ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ιχνηλασία Λειτουργιών Εισόδου/Εξόδου σε Παράλληλα Συστήματα Αρχείων με χρήση Δικτύου Εικονικών Μηχανών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Ε. Ανδριανάκης

Επιβλέπων: Νεκτάριος Κοζύρης
Αναπλ. Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2008

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ιχνηλασία Λειτουργιών Εισόδου/Εξόδου σε Παράλληλα Συστήματα Αρχείων με χρήση Δικτύου Εικονικών Μηχανών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Ε. Ανδριανάκης

Επιβλέπων: Νεκτάριος Κοζύρης
Αναπλ. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Σεπτεμβρίου 2008.

.....	.....	.....
Νεκτάριος Κοζύρης	Γεώργιος Παπακωνσταντίνου	Νικόλαος Παπασπύρου
Αναπλ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2008

.....
Γεώργιος Ε. Ανδριανάκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Γεώργιος Ε. Ανδριανάκης, 2008.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η διπλωματική αυτή πραγματεύεται την ιχνηλασία (tracing) των λειτουργιών εισόδου/εξόδου (Input/Output - I/O) με σκοπό να μελετηθούν ορισμένα χαρακτηριστικά δύο συστημάτων αρχείων. Τα συστήματα αυτά αρχείων είναι τα GPFS και GFS και ανήκουν στις κατηγορίες των Παράλληλων Συστημάτων Αρχείων (Parallel File Systems) και των Συστημάτων Αρχείων Κοινού Μέσου (Shared Disk File Systems) αντίστοιχα. Ένα σημαντικό τμήμα της εργασίας αυτής αποτελεί η δημιουργία μιας υποδομής με χρήση εικονικών μηχανών πάνω στην οποία θα εγκατασταθούν τα συστήματα αρχείων και θα μελετηθούν με χρήση ορισμένων εργαλείων tracing.

Η εργασία αυτή αποτελείται από τρία διακριτά τμήματα. Το πρώτο τμήμα αφορά την ανάπτυξη της υποδομής που είναι απαραίτητη για την εγκατάσταση των υπό μελέτη συστημάτων αρχείων και για να καταστεί δυνατόν εν τέλει, να γίνει η μελέτη στα δύο συστήματα αρχείων. Το στάδιο αυτό περιλαμβάνει την δημιουργία εικονικών μηχανών, την δημιουργία του λοιπού εικονικού hardware καθώς και την εγκατάσταση του λογισμικού των συστημάτων αρχείων πάνω στο εικονικό hardware. Το δεύτερο τμήμα αφορά την ανάλυση των εργαλείων που χρησιμοποιούνται για το tracing των I/O γεγονότων σε όλη την ιεραρχία λογισμικού που εμπλέκεται στην διεξαγωγή του παράλληλου I/O. Το τρίτο και τελευταίο τμήμα αφορά την χρήση της υποδομής που δημιουργήθηκε στο πρώτο τμήμα της εργασίας και ορισμένα από τα εργαλεία tracing που αναλύθηκαν στο δεύτερο τμήμα, προκειμένου να γίνουν ορισμένα πειράματα και να εξαχθούν συμπεράσματα για τη λειτουργία των συστημάτων αρχείων GFS και GPFS. Δίνεται έμφαση κυρίως στην εξαγωγή συμπερασμάτων που αφορούν τον τρόπο χειρισμού του caching των δεδομένων που κάνουν τα δύο αυτά συστήματα αρχείων.

Λέξεις κλειδιά

Παράλληλα Συστήματα Αρχείων, Συστήματα Αρχείων Κοινού Μέσου, Εικονικές Μηχανές, Συστήματα Αποθήκευσης, Λειτουργικό Σύστημα Linux, Ιεραρχία Λογισμικού, Παράλληλου I/O, Tracing, Δίκτυο Αποθηκευτικού Χώρου, GPFS, GFS, blktrace

Abstract

The objective of this study is the use of tracing of Input/Output (I/O) operations in order to study some characteristics of two file systems. These two file systems are GFS and GPFS which are classified as a Parallel File System and a Shared Disk File System respectively. An important part of this study is the creation of an infrastructure based on virtual machines which will be used to install and study the two file systems using tracing tools.

The study is composed of three discrete parts. The first part consists of the creation of the infrastructure necessarily for the installation of the two file systems. Steps included are the creation of virtual nodes that will run the file systems, the rest of the virtual hardware as well as the actual installation of the two file systems on the virtual nodes. The second part consists of the analysis of the tracing tools that are used to trace I/O operations on all levels of the software stack that is involved in parallel I/O. The third and last part presents the use of the infrastructure created in the first part and certain tracing tools analyzed in the second part in order to conduct some experiments and draw some conclusions regarding the two file systems. The main conclusions drawn are ones regarding the way the two file systems handle data caching issues.

Keywords

Parallel File Systems, Shared Disk File Systems, Virtual Machines, Storage Devices, Linux, Software stack, Parallel I/O, Tracing, Storage Area Network, GPFS, GFS, blktrace

Ευχαριστίες

Η συγκεκριμένη διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσοβίου Πολυτεχνείου υπό την επίβλεψη του Αναπληρωτή Καθηγητή, κύριου Νεκτάριου Κοζύρη.

Θα ήθελα πρώτα από όλα να ευχαριστήσω θερμά τον κύριο Κοζύρη που μου έδωσε την ευκαιρία να ασχοληθώ με αυτό το εξαιρετικά ενδιαφέρον και σύγχρονο θέμα καθώς και για την αμέριστη στήριξη και βοήθεια που μου παρείχε κατά την πορεία της εκπόνησης της εργασίας αυτής.

Θέλω ιδιαίτερα να ευχαριστήσω τους υποψήφιους διδάκτορες Βαγγέλη Κούκη και Τάσο Νάνο, που χωρίς την άμεση επικοινωνία και την συνεχή παροχή εκ μέρους τους, τεχνικής βοήθειας και ιδεών για να ξεπεραστούν οι σκόπελοι που προέκυπταν, δεν θα είχε ολοκληρωθεί αυτή η εργασία.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Εισαγωγή …… 15

1. Συστήματα Αποθήκευσης …….. 17
1.1 Εισαγωγή …… 17
1.2 Σκληρός Δίσκος …… ……. 17
1.2.1 Cache Σκληρού Δίσκου …… 19
1.3 Συστοιχίες Σκληρών Δισκων (Redundant Array of Inexpensive Storage, RAID) …….……. 20
1.4 Αρχιτεκτονική I/O Συστημάτων …… 24
1.4.1 Ανταλλαγή δεδομένων μεταξύ συσκευών και κεντρικής μνήμης … 26
1.5 Τεχνολογίες Δικτυακής Αποθήκευσης ……... 28
1.5.1 NAS (Network Attached Storage) …… 28
1.5.2 SAN (Storage Area Network) …….…….. 29
1.5.3 Διαφορές SAN – NAS …….. 30
1.5.4 Επιπτώσεις της αρχιτεκτονικής των SAN και NAS στην επεκτασιμότητα και την ταχύτητα …… 32
1.5.5 Θέματα Caching στα SAN και NAS …… 34
1.5.6 Τεχνολογία Fibre Channel …… 35

2. Γενικά Χαρακτηριστικά Αρχείων και Συστημάτων Αρχείων…… 39
2.1 Εισαγωγή …….…… 39
2.2 Αρχεία (Files) ……. 39
2.3 Συστήματα Αρχείων (File Systems)…….…… 43
2.4 Γενικά για το Caching Αρχείων …….. 45

3. Συστήματα Αποθήκευσης στο Linux …… 49
3.1 Εισαγωγή ……. 49
3.2 Γενικά για το Linux ……. 49
3.3 Linux VFS …… 51
3.4 Αρχεία Συσκευών στο Linux …… 54
3.5 Χειρισμός Συσκευών Block στο Linux …… 56
3.6 Linux και Caching Αρχείων ……. 59

4. Παράλληλα και Κοινού Μέσου Συστήματα Αρχείων …… 61
4.1 Εισαγωγή ……. 61
4.2 Συστήματα Αρχείων Κοινού Μέσου …….. 61
4.3 Κατανεμημένα Συστήματα Αρχείων ……... 65
4.4 Παράλληλα Συστήματα Αρχείων ……. 65
4.5 Διαφορά Κατανεμημένων και Συστημάτων Αρχείων Κοινού Μέσου …. 66
4.6 Διαφορά στόχων Παράλληλων Συστημάτων Αρχείων και Συστημάτων Αρχείων Κοινού Μέσου …… 66
4.7 Παραδείγματα Παράλληλων Συστημάτων Αρχείων και Συστημάτων Αρχείων Κοινού Μέσου …… 67
4.7.1 GPFS ……..... 67
4.7.2 GFS ……..... 68
4.7.3 OCFS2 ……. 69
4.7.4 PVFS2 …….. 69
4.8 Διαφορές GFS και GPFS …….. 70

4.9 Ομοιότητες GFS και GPFS …… 80

5. Τύποι Παράλληλων Υπολογιστικών Συστημάτων ……..... 83
5.1 Εισαγωγή …….. 83
5.2 Πολυπύρηνα Συστήματα (Multicore Systems) …… 83
5.3 Συμμετρικά Συστήματα Πολυεπεξεργασίας (Symmetric Multiprocessing Systems) …….……. 83
5.4 Massively Parallel Processing (MPP) Systems …… 84
5.5 Συστοιχίες Υπολογιστών (Computer Clusters) …… 84
5.6 Υπολογιστικά Πλέγματα (Grid Computing) ……. 86

6. Υποδομή και Εγκατάσταση ……..…..... 89
6.1 Εισαγωγή ……..…. 89
6.2 Δομή Συστήματος ……..…… 89
6.3 Virtualization …….. 90
6.3.1 Xen …… 92
6.4 Πλεονεκτήματα και Μειονεκτήματα της Χρήσης Εικονικού Hardware … 93
6.5 Διαδικασία Εγκατάστασης …….. 94
6.5.1 Πορεία μέχρι την τελική εγκατάσταση ……. 94
6.5.2 Τελική Εγκατάσταση και δημιουργία των Συστημάτων Αρχείων …… 97

7. Τεχνικές Καταγραφών – Tracing …….. 103
7.1 Εισαγωγή …… 103
7.2 Λόγοι χρησιμοποίησης I/O Tracing ..…… 103
7.3 Tracing στην Ιεραρχία Λογισμικού για Παράλληλο I/O …… 103
7.4 blktrace …….…….. 110
7.5 GPFS Tracing..…..…… 112

8. Πειραματικά Συμπεράσματα για τα GFS και GPFS …….. 113
8.1 Εισαγωγή …… 113
8.2 Λογισμικό ……..… 113
8.2.1 mBench ……. 113
8.2.2 mWrite …….. 114
8.2.3 gpfsperf ……. 114
8.2.4 BTIO ……. 114
8.2.5 Προγράμματα comparer …… 115
8.3 Πειράματα και Συμπεράσματα …… 116
8.3.1 Πείραμα 1 …….. 116
8.3.2 Πείραμα 2 …….. 118
8.3.3 Πείραμα 3 …….. 120
8.3.4 Πείραμα 4 …….. 127
8.3.5 Πείραμα 5 …….. 131

9. Επίλογος …….. 139
9.1 Σύνοψη ……. 139
9.2 Επεκτάσεις…….. 139

Βιβλιογραφία ……... 141

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Σχήμα 1.1 - Σκληρός Δίσκος …….. 18
Σχήμα 1.2 - Ιεραρχία Μνήμης …… 19
Σχήμα 1.3 - RAID 3 …… 21
Σχήμα 1.4 - RAID 5 …… 22
Σχήμα 1.5 - RAID 6 …… 23
Σχήμα 1.6 - RAID 0+1 …… 24
Σχήμα 1.7 - RAID 1+0 …… 24
Σχήμα 1.8 - Αρχιτεκτονικη PC - Διάδρομοι (Buses) …….. 25
Σχήμα 1.9 - NAS…….. …….. 28
Σχήμα 1.10 - SAN…….. …….. 29
Σχήμα 1.11 - Διαφορά μοντέλου αποθήκευσης σε SAN και NAS ……...…….. 32
Σχήμα 1.12 - Τοπολογίες Fibre Channel ……...…….. 36
Σχήμα 3.1 - Παράδειγμα χρήσης VFS ……. 52
Σχήμα 3.2 - Χρήση δομών δεδομένων του VFS ……. 54
Σχήμα 3.3 - Λογισμικό του πυρήνα που εμπλέκεται στον χειρισμό block συσκευών ……...……... 56
Σχήμα 4.1 - Απευθείας πρόσβαση στον αποθηκευτικό χώρο ……. 62
Σχήμα 4.2 - NBD πρόσβαση στον αποθηκευτικό χώρο …… 62
Σχήμα 7.1 - Ιεραρχία Λογισμικού για Παράλληλο IO …… 104
Σχήμα 7.2 - TraceFS …… 109
Σχήμα 8.1 - Σειριακή Ανάγνωση Αρχείου σε Τέταρτα – GPFS …… 122
Σχήμα 8.2 - Σειριακή Ανάγνωση Αρχείου σε Τέταρτα – GFS ……... 123
Σχήμα 8.3 - Σειριακή Ανάγνωση Αρχείου σε Μισά – GPFS …….. 123
Σχήμα 8.4 - Σειριακή Ανάγνωση Αρχείου σε Μισά – GFS …… 124
Σχήμα 8.5 - Αντίστροφη Σειριακή Ανάγνωση Αρχείου σε Τέταρτα – GPFS … 124
Σχήμα 8.6 - Αντίστροφη Σειριακή Ανάγνωση Αρχείου σε Τέταρτα – GFS ….. 125
Σχήμα 8.7 - Αντίστροφη Σειριακή Ανάγνωση Αρχείου σε Μισά - GPFS ……. 125
Σχήμα 8.8 - Αντίστροφη Σειριακή Ανάγνωση Αρχείου σε Μισά - GFS …… 126
Σχήμα 8.9 - Blocks που διάβαζουν οι κόμβοι όταν δεν υπάρχει DS …… 136
Σχήμα 8.10 -Blocks που διάβαζουν οι κόμβοι όταν υπάρχει DS – Καθόλου λεπτομέρεια …….. 137
Σχήμα 8.11 -Blocks που διάβαζουν οι κόμβοι όταν υπάρχει DS – Μικρή λεπτομέρεια ……. 137
Σχήμα 8.12 -Blocks που διάβαζουν οι κόμβοι όταν υπάρχει DS – Μεγάλη λεπτομέρεια …… 138

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1.1 - Σύγκριση Fibre Channel, Gigabin Ethernet, ATM …….. 37
Πίνακας 3.1 - Παραδείγματα αρχείων συσκευών……. 55
Πίνακας 8.1 – Πείραμα 3, Αρχείο 50MB …… 121
Πίνακας 8.2 - Πείραμα 3, Αρχείο 75MB …… 121
Πίνακας 8.3 - Πείραμα 3, Αρχείο 100MB …… 122
Πίνακας 8.4 - Πείραμα 5, GFS, κλάση S ……. 133
Πίνακας 8.5 - Πείραμα 5, GFS, κλάση W……. 134
Πίνακας 8.6 - Πείραμα 5, GPFS, κλάση S …… 134
Πίνακας 8.7 - Πείραμα 5, GPFS, κλάση W …….. 134
Πίνακας 8.8 - Πείραμα 5, Data Shipping ……. 135

Εισαγωγή

Στόχος αυτής της διπλωματικής είναι να δημιουργηθεί μια υποδομή που θα επιτρέψει την εγκατάσταση των συστημάτων αρχείων GFS και GPFS σε εικονικό hardware, ν' αναλυθούν οι τεχνικές tracing και τέλος με χρήση ορισμένων εργαλείων tracing, να εξαχθούν ορισμένα συμπεράσματα που αφορούν τον τρόπο λειτουργίας των δύο αυτών συστημάτων αρχείων.

Στο Κεφάλαιο 1 παρουσιάζονται συνοπτικά τα συστήματα αποθήκευσης. Εκτός από τον σκληρό δίσκο που αποτελεί το βασικό σύστημα αποθήκευσης, παρουσιάζονται και τα δικτυακά συστήματα αποθήκευσης SAN και NAS. Επίσης αναλύονται και οι μέθοδοι που χρησιμοποιεί το λειτουργικό σύστημα για να επιτυγχάνει την πρόσβαση στα αποθηκευτικά μέσα.

Στο κεφάλαιο 2 παρουσιάζονται τα βασικά θεωρητικά στοιχεία που αφορούν τον χειρισμό των αρχείων και των συστημάτων αρχείων από τα σύγχρονα λειτουργικά συστήματα.

Στο κεφάλαιο 3 αναλύεται ο χειρισμός των συστημάτων αποθήκευσης που κάνει το λειτουργικό σύστημα Linux.

Στο Κεφάλαιο 4 εισάγονται οι έννοιες του παράλληλου συστήματος αρχείων και του συστήματος αρχείων κοινού μέσου. Στη συνέχεια αναλύονται και συγκρίνονται τα συστήματα αρχείων GPFS και GFS.

Στο Κεφάλαιο 5 παρουσιάζονται συνοπτικά τα είδη των παράλληλων συστημάτων που χρησιμοποιούνται σήμερα. Μεγαλύτερη βαρύτητα δίνεται στην περιγραφή του Cluster, που είναι και ο τύπος του παράλληλου συστήματος που χρησιμοποιείται (και προσομοιώνεται) στην διπλωματική αυτή.

Στο κεφάλαιο 6 αναλύεται η δομή του συστήματος -της υποδομής- που χρησιμοποιείται για την μελέτη μας και περιγράφεται συνοπτικά η διαδικασία εγκατάστασης των συστημάτων αρχείων GFS και GPFS.

Στο Κεφάλαιο 7 περιγράφονται αναλυτικά οι τεχνικές tracing που μπορούν να εφαρμοστούν σε όλα τα επίπεδα της ιεραρχίας του λογισμικού που χρησιμοποιείται στο παράλληλο I/O.

Στο Κεφάλαιο 8 χρησιμοποιούνται η υποδομή των εικονικών μηχανών και ορισμένα από εργαλεία για την διεξαγωγή πειραμάτων πάνω στα συστήματα αρχείων GFS και GPFS. Τα πειράματα αυτά μας δίνουν την δυνατότητα να εξάγουμε ορισμένα συμπεράσματα για τον τρόπο λειτουργίας των υπό μελέτη συστημάτων αρχείων.

Τέλος στο Κεφάλαιο 9 δίνεται μια σύνοψη των προηγούμενων κεφαλαίων και αναφέρονται πιθανές επεκτάσεις της μελέτη αυτής.

Κεφάλαιο 1.
Συστήματα Αποθήκευσης

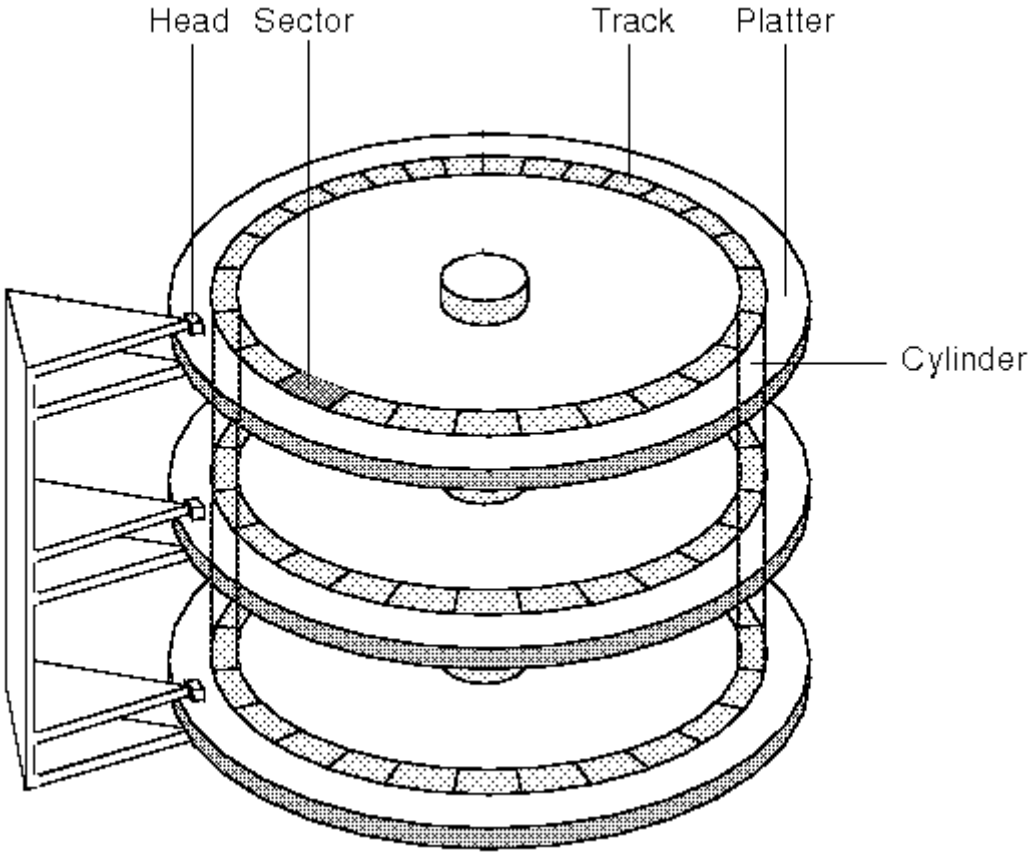
1.1 Εισαγωγή

Οι δύο κύριες εργασίες που έχει να επιτελέσει ένα υπολογιστικό σύστημα είναι η επεξεργασία δεδομένων και το I/O (Είσοδος/Εξοδος, Input/Output). Σε ορισμένες περιπτώσεις μάλιστα, η κύρια εργασία είναι το I/O ενώ η επεξεργασία συμβαίνει περιστασιακά ως αποτέλεσμα κάποιας I/O ενέργειας.
Παρά την σπουδαιότητά τους, οι επιδόσεις των I/O συστημάτων υπολείπονται σημαντικά εκείνων των επεξεργαστών και της κεντρική μνήμης με αποτέλεσμα τα I/O συστήματα να αποτελούν σημείο συμφόρησης (bottleneck) των συνολικών επιδόσεις των υπολογιστικών συστημάτων.

Η πιο σημαντική κατηγορία I/O συστημάτων είναι τα συστήματα αποθήκευσης. Στα συστήματα αποθήκευσης, η αποθήκευση είναι μόνιμη, δηλαδή τα δεδομένα παραμένουν ακόμα και όταν ο υπολογιστής είναι κλειστός.
Η κατηγορία των συστημάτων αποθήκευσης περιλαμβάνει πολλών ειδών συσκευές. Μερικές από αυτές είναι οι σκληροί δίσκοι, οι δισκέτες, οι οπτικοί δίσκοι καθώς και οι συσκευές σταθερής κατάστασης ημιαγώγιμης μνήμης, γνωστές ως συσκευές flash. Στην συνέχεια θα περιγράψουμε τον σκληρό δίσκο που αποτελεί το πιο ευρέως χρησιμοποιούμενο σύστημα αποθήκευσης. Αργότερα θα αναφερθούμε στα δικτυακά συστήματα αποθήκευσης.

1.2 Σκληρός Δίσκος

Ο σκληρός δίσκος αποτελεί το κύριο σύστημα αποθήκευσης σε σύγχρονα υπολογιστικά συστήματα. Η αρχιτεκτονική του σκληρού δίσκου είναι σχετικά απλή. Αποτελείται από ορισμένες (συνήθως από μία έως τέσσερις) λεπτές κυκλικές πλάκες (platters) όμοιες στην όψη με cd που περιστρέφονται και μια κεφαλή ανάγνωσης / εγγραφής για κάθε πλάκα.
Οι πλάκες κατασκευάζονται από μη μαγνητικό υλικό (όπως κράμα αλουμινίου ή γυαλί) και καλύπτονται και από τις δύο πλευρές (surfaces) τους με μια ειδική επίστρωση μαγνητικού υλικού (όπως κράμα κοβάλτιου). Χρησιμοποιώντας αυτό το μαγνητικό υλικό αποθηκεύονται τα δεδομένα στον σκληρό δίσκο. Οι πλάκες αυτές συνήθως έχουν διάμετρο από 1.8 έως 3.5 ίντσες και περιστρέφονται με σταθερή γωνιακή ταχύτητα από 5.400 έως 15000 στροφές το λεπτό ανάλογα με τον δίσκο. Η επιφάνεια των πλακών χωρίζεται σε ομόκεντρους κύκλους που ονομάζονται ίχνη (tracks). Συνήθως υπάρχουν από 10000 έως 50000 ίχνη ανά πλάκα. Με τη σειρά τους τα ίχνη χωρίζονται σε τομείς (sectors). Συνήθως κάθε ίχνος αποτελείται από 100 έως 500 τομείς. Οι τομείς αυτοί αποθηκεύουν τα δεδομένα. Το πιο σύνηθες μέγεθος τομέα είναι 512 bytes και αποτελεί την μικρότερη δυνατή μονάδα που μπορεί να μεταφερθεί από ή προς τον σκληρό δίσκο.
Η δομή του σκληρού δίσκου φαίνεται στην παρακάτω εικόνα.



Σχήμα 1.1: Σκληρός Δίσκος

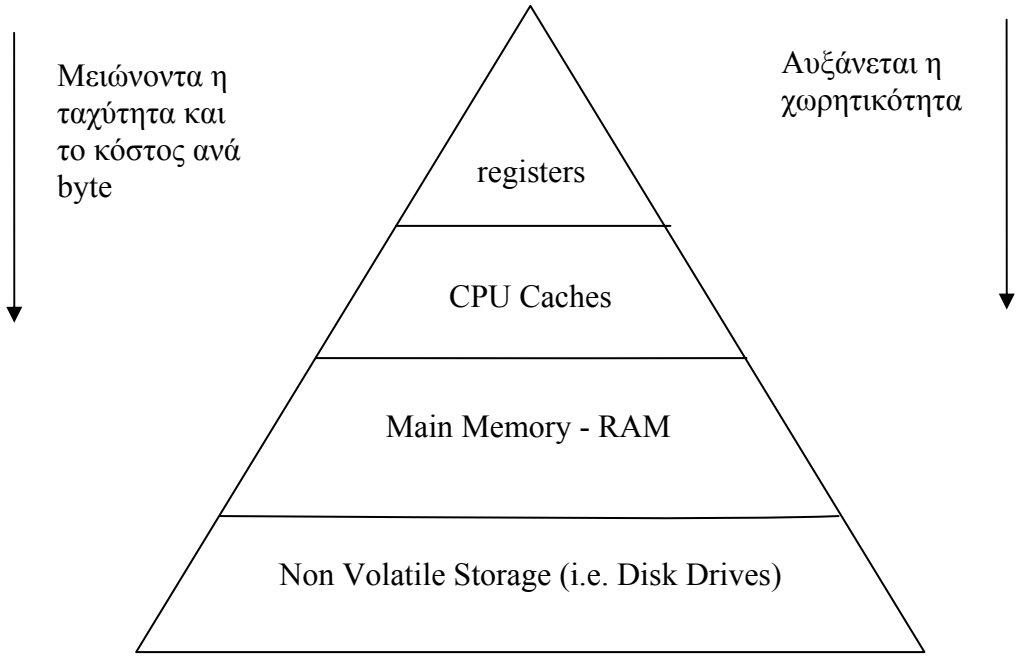
Οι κεφαλές ανάγνωσης / εγγραφής αποτελούνται από μαγνητικά ευαίσθητα υλικά και μπορούν να διαβάσουν τα δεδομένα που έχουν γραφτεί στις μαγνητικές πλάκες. Οι κεφαλές βρίσκονται όλες πάντα στο ίδιο κάθετο επίπεδο και ενώνονται η καθεμία μέσω ενός βραχίονα με ένα σταθερό σημείο σύνδεσης (όπως φαίνεται και στην εικόνα). Το σύνολο των ιχνών που βρίσκονται κάτω από τις κεφαλές ανάγνωσης ονομάζεται κύλινδρος.
Πρέπει να τονίσουμε ότι οι κεφαλές βρίσκονται σε πάρα πολύ μικρή απόσταση από τις πλάκες (σε νεότερους δίσκους η απόσταση αυτή είναι της τάξης των δεκάδων νανόμετρων) και μπορούν να καταστρέψουν μια πλάκα αν έρθουν σε επαφή μαζί της. Συνεπώς απαιτείται πολύ μεγάλη προσοχή στον χειρισμό των σκληρών δίσκων.

Για την προσπέλαση των δεδομένων του δίσκου, το λειτουργικό σύστημα πρέπει να καθοδηγήσει τον δίσκο μέσα από μια διαδικασία τριών σταδίων. Το λειτουργικό σύστημα (μέσω ενός οδηγού, driver) στέλνει στον σκληρό δίσκο τις απαραίτητες εντολές για την προσπέλαση των δεδομένων και ένας ειδικού σκοπού επεξεργαστής που βρίσκεται στον σκληρό δίσκο (ο ελεγκτής του σκληρού δίσκου) μεταφράζει τις εντολές αυτές σε ηλεκτρονικές και μηχανικές ενέργειες του δίσκου. Σημειώνουμε ότι η μεταφορά δεδομένων από και προς τον σκληρό δίσκο γίνεται σε αιτήσεις που περιλαμβάνουν πολλαπλούς γειτονικούς τομείς.
Το πρώτο στάδιο είναι η τοποθέτηση της κεφαλής ανάγνωσης / εγγραφής πάνω από το ίχνος που περιλαμβάνει τα προς μεταφορά δεδομένα. Το στάδιο αυτό ονομάζεται αναζήτηση (seek) και ο χρόνος που χρειάζεται για να πραγματοποιηθεί ονομάζεται

χρόνος αναζήτησης (seek time).
Το δεύτερο στάδιο είναι η μετακίνηση του πρώτου τομέα των δεδομένων κάτω από την κεφαλή. Κατά την διαδικασία αυτή η κεφαλή δεν μετακινείται, απλώς περιμένει να βρεθεί από κάτω της ο τομέας μέσω της περιστροφής των πλακών. Ο χρόνος που περνάει για να γίνει αυτή η διαδικασία ονομάζεται χρόνος περιστροφής (rotation latency).
Το τελευταίο στάδιο είναι η ανάγνωση ή εγγραφή των τομέων που αντιστοιχούν στα προς μεταφορά δεδομένα. Ο χρόνος που καταλαμβάνει αυτή η διαδικασία ονομάζεται χρόνος μεταφοράς (transfer time).

1.2.1 Cache Σκληρού Δίσκου

Στην παραπάνω περιγραφή των τριών σταδίων μεταφοράς δεδομένων από η προς τον σκληρό δίσκο, δεν λάβαμε υπόψιν καθόλου την ύπαρξη της cache του σκληρού δίσκου. Όλοι οι σύγχρονοι σκληροί δίσκοι διαθέτουν δικές τους cache που σκοπό έχουν την ταχύτερη πρόσβαση στα δεδομένα του δίσκου. Η cache του δίσκου συνήθως δεν θεωρείται μέρος της ιεραρχίας της μνήμης όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 1.2: Ιεραρχία Μνήμης

Ο τρόπος με τον οποίο η cache του δίσκου αυξάνει τις επιδόσεις του σκληρού δίσκου είναι με τη μείωση των προσβάσεων που απαιτούνται στο μαγνητικό μέσο του σκληρού δίσκου. Αυτό γίνεται με δύο τρόπους: Πρώτον με το να διατηρεί η cache δεδομένα που αναγνώστηκαν πρόσφατα από τον δίσκο, με την ελπίδα ότι τα δεδομένα αυτά θα διαβαστούν εκ νέου στη συνέχεια. Προσπαθεί δηλαδή η cache να

εκμεταλλευτεί την χρονική τοπικότητα των προσπελάσεων των δεδομένων που κατά κανόνα ισχύει σε ένα υπολογιστικό σύστημα. Δεύτερον με το να εισέρχονται στην cache περισσότερα δεδομένα από αυτά που απαιτούνται σε μια αίτηση ανάγνωσης του δίσκου, με την ελπίδα ότι θα υπάρξει σύντομα νέα αίτηση ανάγνωσης που θα θέλει τα επόμενα στον δίσκο δεδομένα από αυτά που ζήτησε η προηγούμενη ανάγνωση.
Η διαδικασία ανάγνωσης περισσότερων δεδομένων από αυτών που απαιτούνται ονομάζεται prefetching. Στην περίπτωση που περιγράφουμε ελέγχεται από τον ελεγκτή του σκληρού δίσκου και τα επιπλέον δεδομένα (όπως και αυτά που απαιτούνται από την αίτηση προσπέλασης) μπαίνουν στην cache του σκληρού δίσκου. Το prefetching αυξάνει τις επιδόσεις του σκληρού δίσκου όταν δεδομένα του δίσκου διαβάζονται σειριακά, που είναι μια τυπική περίπτωση πρόσβασης στα δεδομένα ενός δίσκου.

1.3 Συστοιχία Σκληρών Δίσκων (Redundant Arrays of Inexpensive Disks, RAID)

Με το χαμηλό κόστος που έχουν σήμερα, είναι εφικτή η τοποθέτηση πολλών σκληρών δίσκων σε ένα υπολογιστικό σύστημα. Με την χρήση πολλών σκληρών δίσκων δίνεται η δυνατότητα όχι μόνο να αυξηθεί ο χώρος αποθήκευσης, αλλά και να αυξηθούν οι I/O επιδόσεις και η προστασία των δεδομένων. Οι υψηλότερες επιδόσεις επιτυγχάνονται αν πολλαπλοί δίσκοι λειτουργούν παράλληλα, ενώ η προστασία των δεδομένων αυξάνεται αν δεδομένα που υπάρχουν σε έναν δίσκο αντιγράφονται σε άλλους. Το RAID αποτελεί ένα πρότυπο που δίνει την δυνατότητα της εκμετάλλευσης αυτών των δυνατοτήτων.

Παλαιότερα το πρότυπο RAID χρησιμοποιούνταν ως μια φθηνή εναλλακτική των μεγάλων και ακριβών σκληρών δίσκων, αλλά στη σημερινή εποχή των φθηνών σκληρών δίσκων χρησιμοποιείται ως ένα μέσο για αύξηση των επιδόσεων ή/και της ασφάλειας των δεδομένων.

Με τη χρήση του προτύπου RAID είναι εφικτή η ομαδοποίηση των σκληρών δίσκων σε μια ενιαία λογική μονάδα αποθήκευσης. Όταν γίνεται αυτή η ομαδοποίηση το λειτουργικό σύστημα (καθώς και οι εφαρμογές επιπέδου χρήστη) αντιλαμβάνονται την ύπαρξη μιας μόνο συσκευής αποθήκευσης αντί για πολλές. Η ομαδοποίηση μπορεί να γίνει είτε μέσω της χρήσης ειδικών hardware ελεγκτών είτε μέσω της χρήσης ειδικού λογισμικού.
Υπάρχουν διάφοροι τρόποι μέσω των οποίων μπορεί να γίνει η κατανομή των δεδομένων στους δίσκους με τη χρήση του RAID και ο καθένας εξυπηρετεί ένα συγκεκριμένο σκοπό.
Παρακάτω παραθέτουμε εν συντομία αυτούς τους τρόπους.

RAID 0

Το RAID 0 χρησιμοποιεί δύο ή περισσότερους δίσκους για να κατανείμει δεδομένα αρχείων. Με αυτό τον τρόπο αυξάνεται αθροιστικά με τον αριθμό των δίσκων ο ρυθμός διαμεταγωγής σε κάθε αρχείο (μέχρι το όριο που επιβάλλει η ύπαρξη του

ελεγκτή RAID είτε αυτός είναι hardware ή software) αφού μπορούν να γίνουν προσβάσεις στους δίσκους που τα περιέχουν, εν παραλλήλω. Επειδή όμως τα δεδομένα των αρχείων κατανέμονται σε όλους τους δίσκους, η απώλεια ενός δίσκου, σημαίνει και καταστροφή των δεδομένων όλων των δίσκων. Ο συνολικός χώρος που φαίνεται στο λειτουργικό σύστημα είναι το άθροισμα του χώρου των δίσκων άρα δεν υπάρχει απώλεια αποθηκευτικού χώρου.

RAID 1

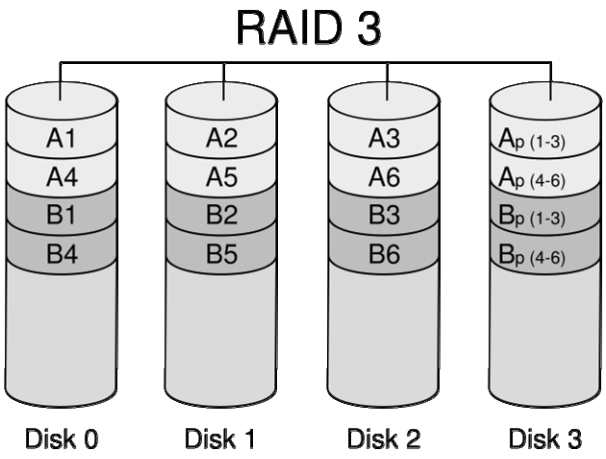
Για την χρησιμοποίηση της διάταξης RAID 1 είναι απαραίτητη η ύπαρξη δύο ή περισσοτέρων δίσκων ιδίου μεγέθους. Τα δεδομένα που γράφονται στον ένα δίσκο αντιγράφονται αυτόματα και στους άλλους δίσκους, προσφέροντας ασφάλεια δεδομένων αφού σε περίπτωση απώλειας ενός δίσκου, μπορεί να χρησιμοποιηθεί ένας εκ των υπολοίπων. Το μειονέκτημα της διάταξης αυτής είναι ότι η συνολική χωρητικότητα του λογικού δίσκου που αντιλαμβάνεται το λειτουργικό σύστημα είναι η χωρητικότητα ενός μόνο δίσκου.

RAID 2

Το RAID 2 παρέχει ασφάλεια δεδομένων μέσω του κώδικα διόρθωσης λαθών Hamming. Τα δεδομένα των αρχείων μοιράζονται σε όλους τους δίσκους σε τμήματα μεγέθους ενός byte. Σπάνια χρησιμοποιείται στην πράξη αφού ο αριθμός των πλεοναζόντων δίσκων είναι λογαριθμικός του συνολικού αριθμού των δίσκων και δεν παρέχει υψηλές επιδόσεις.

RAID 3

Αυτή η διάταξη χρησιμοποιεί τουλάχιστον τρεις δίσκους. Τα δεδομένα των αρχείων μοιράζονται σε τμήματα μεγέθους ενός byte σε όλους τους δίσκους, εκτός από έναν που αποκλειστικός σκοπό έχει την διατήρηση πληροφοριών για τον έλεγχο ισοτιμίας και δεν λογίζεται ο αποθηκευτικός χώρος του ως επιπλέον χώρος της διάταξης.
Ο τρόπος που κατανέμονται τα δεδομένα φαίνεται στην παρακάτω εικόνα.



Σχήμα 1.3: RAID 3

Βλέπουμε στην εικόνα ότι οι τρεις πρώτοι δίσκοι αποθηκεύουν συνεχόμενα δεδομένα αρχείων, ενώ ο τέταρτος δίσκος διατηρεί πληροφορίες ισοτημίας για κάθε διαμοιρασμό.

Με την διάταξη αυτή, αν ο ένας από τους δίσκους στους οποίους έχουν μοιραστεί τα δεδομένα, χαλάσει, τα δεδομένα δεν χάνονται. Αν χαλάσει ο δίσκος που διατηρεί την ισοτιμία, η διάταξη συνεχίζει να λειτουργεί αλλά χωρίς έλεγχο ισοτιμίας οπότε ουσιαστικά εκφυλίζεται σε RAID 0.

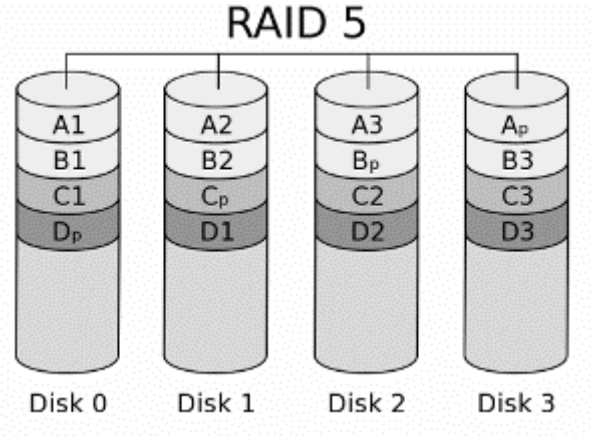
Με τον διαμοιρασμό των δεδομένων των αρχείων σε πολλούς δίσκους επιτυγχάνεται υψηλός ρυθμός διαμεταγωγής κατά την ανάγνωση δεδομένων αφού τα δεδομένα κάθε αρχείου διαβάζονται παράλληλα από πολλούς δίσκους. Δυστυχώς στην περίπτωση της εγγραφής ο διαμοιρασμός των δεδομένων σε πολλούς δίσκους δεν αρκεί για ανάλογη αύξηση των επιδόσεων, καθώς η εγγραφή των πληροφοριών ισοτιμίας σε έναν δίσκο αποτελεί bottleneck.

RAID 4

Η μόνη διαφορά αυτής της διάταξης με την προηγούμενη είναι ότι τα δεδομένα διαμοιράζονται στους δίσκους σε τμήματα μεγέθους ενός block αντί ενός byte.

RAID 5

Στο RAID 5 χρησιμοποιούνται ο διαμοιρασμός δεδομένων σε δίσκους σε τμήματα block και η εγγραφή πληροφοριών ισοτιμίας όπως και στο RAID 4 με την διαφορά να είναι ότι οι πληροφορίες ισοτιμίας είναι κατανεμημένες σε όλους τους δίσκους. Δηλαδή οι πληροφορίες ισοτιμίας για κάθε διαμοιρασμό δεν αποθηκεύονται σε έναν συγκεκριμένο δίσκο, αλλά τυχαία σε κάποιον από τους δίσκους της διάταξης. Με αυτό τον τρόπο κάθε δίσκος περιέχει πληροφορίες ισοτιμίες για ορισμένους διαμοιρασμούς, αλλά όλοι οι δίσκοι μαζί περιλαμβάνουν όλες τις απαραίτητες πληροφορίες ισοτιμίας.
Για την καλύτερη κατανόηση της κατανομής των δεδομένων και των πληροφοριών ισοτιμίας, παρατίθεται η παρακάτω εικόνα.



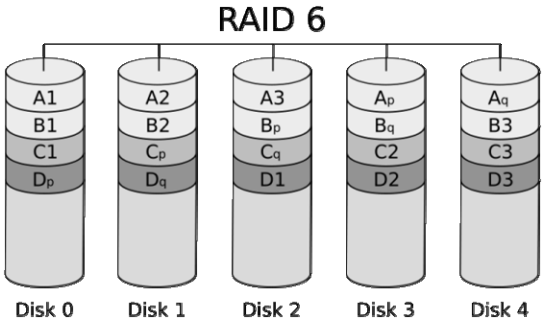
Σχήμα 1.4: RAID 5

Η διάταξη αυτή δίνει αυξημένες επιδόσεις λόγω παράλληλων προσβάσεων στους δίσκους και στην περίπτωση της ανάγνωσης και στην περίπτωση της εγγραφής δεδομένων. Επιπλέον απαιτεί τουλάχιστον τρεις δίσκους για να λειτουργήσει και μπορεί να συνεχίσει να λειτουργεί στην περίπτωση που ένας δίσκος καταστραφεί, αφού τα δεδομένα του δίσκου μπορούν να ανακατασκευαστούν από τα δεδομένα των άλλων. Αν καταστραφεί και δεύτερος, τότε έχουμε συνολική απώλεια δεδομένων.

Σε αυτή την διάταξη δεν υπάρχουν πλεονάζοντες δίσκοι. Μέρος όμως του αποθηκευτικού χώρου κάθε δίσκου χρησιμοποιείται για την διατήρηση των πληροφοριών
ισοτιμίας, συνεπώς ο συνολικός αποθηκευτικός χώρος της διάταξης είναι μικρότερος του αθροίσματος των χωρητικοτήτων των δίσκων.

RAID 6

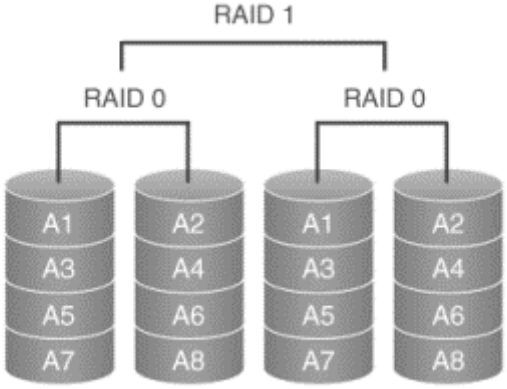
Η διάταξη αυτή είναι παρόμοια με την προηγούμενη αλλά μπορεί να προστατέψει τα δεδομένα από απώλεια δύο δίσκων. Αυτό γίνεται κρατώντας τις πληροφορίες ισοτιμίας για κάθε διαμοιρασμό, σε δύο δίσκους ταυτόχρονα. Η διάταξη φαίνεται στην παρακάτω εικόνα.



Σχήμα 1.5: RAID 6

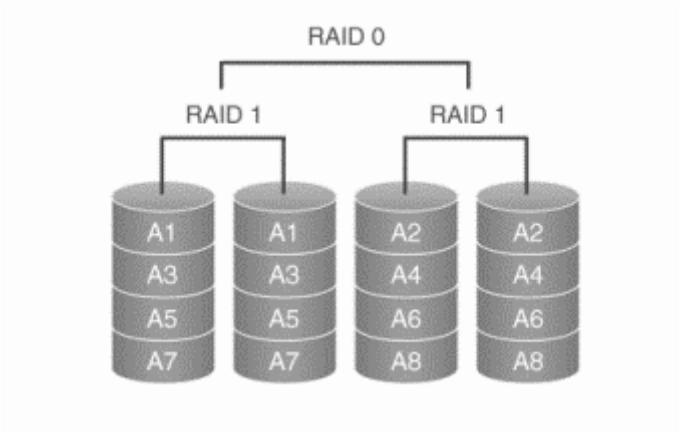
Ο ελάχιστος αριθμός δίσκων για μια διάταξη RAID 6 είναι τέσσερις. Η επιβάρυνση του συνολικού αποθηκευτικού χώρου με πληροφορίες ισοτιμίας είναι διπλάσια από ότι στο RAID 5.

Στην πράξη οι πιο ευρέως χρησιμοποιούμενες διατάξεις είναι οι RAID 0, RAID 1 και RAID 5. Πολλές φορές όμως απαντώνται και υβριδικές διατάξεις που επιτυγχάνουν τον απαραίτητο από συγκεκριμένες εφαρμογές συνδυασμό επιδόσεων και ασφάλειας δεδομένων. Οι πιο κοινές τέτοιες διατάξεις είναι οι RAID 0+1 και RAID 1+0 που φαίνονται στις παρακάτω εικόνες.



Σχήμα 1.6: RAID 0+1

Σε αυτή την εικόνα βλέπουμε πως το κάτω επίπεδο του RAID είναι το RAID 0, εφόσον γίνεται διαμοιρασμός συνεχόμενων δεδομένων στους δύο πρώτους δίσκους. Στη συνέχεια χρησιμοποιείται RAID 1 προκειμένου να αντιγραφούν τα περιεχόμενα της πρώτης συστοιχίας RAID 0, στην δεύτερη.

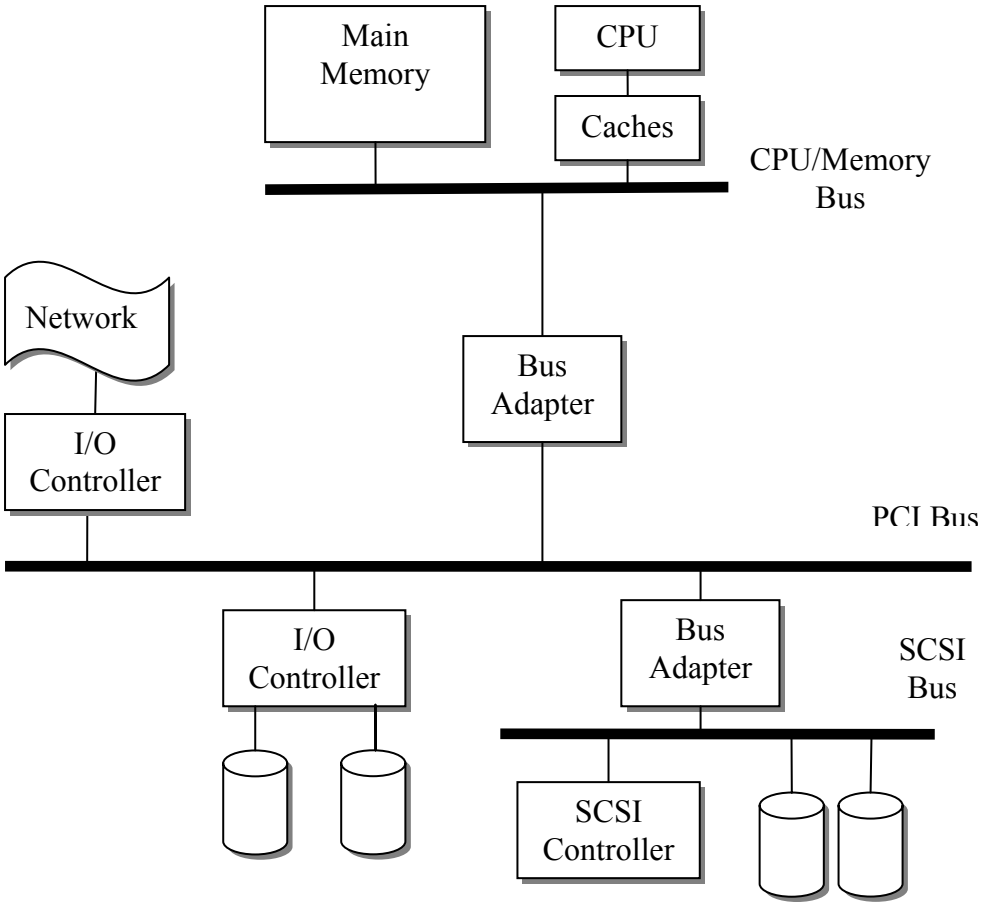


Σχήμα 1.7: RAID 1+0

Στην εικόνα αυτή βλέπουμε πως αυτή η διάταξη είναι μια ανεστραμμένη μορφή της προηγούμενης. Στο κατώτερο χρησιμοποιείται RAID 1 για να αντιγραφούν τα περιεχόμενα του πρώτου δίσκου στον δεύτερο, αλλά στο ανώτερο επίπεδο χρησιμοποιείται RAID 0 ώστε οι δύο συστοιχίες RAID 1 να μην περιέχουν τα ίδια δεδομένα.

1.4 Αρχιτεκτονική I/O Συστημάτων

Μέσα σε έναν υπολογιστή υπάρχουν πολλά συστήματα που πρέπει να επικοινωνήσουν μεταξύ τους, με τον επεξεργαστή και με την μνήμη. Η επικοινωνία όλων αυτών των συστημάτων γίνεται συνήθως με την χρήση ενός διαδρόμου (bus). Ο διάδρομος είναι κοινό μέσο επικοινωνίας που αποτελείται από ένα σύνολο καλωδιώσεων που συνδέουν μεταξύ τους τα συστήματα του υπολογιστή. Μια εικόνα μιας τυπικής αρχιτεκτονικής υπολογιστή όπου χρησιμοποιείται διάδρομος για την επικοινωνία των συστημάτων είναι η παρακάτω:



Σχήμα 1.8: Αρχιτεκτονική PC - Διάδρομοι (Buses)

Υπάρχουν δύο πλεονεκτήματα που συνεπάγεται η χρήση του διαδρόμου. Το πρώτο είναι η προσαρμοστικότητα, καθώς με τον ορισμό ενός μοναδικού σχήματος επικοινωνίας, μπορούν πολύ εύκολα συσκευές να προστεθούν και να αφαιρεθούν από τον διάδρομο καθώς και να μετακινηθούν από έναν υπολογιστή σε έναν άλλο που ακολουθεί την ίδια αρχιτεκτονική διαδρόμου. Το δεύτερο πλεονέκτημα είναι η μείωση του κόστους, καθώς με ένα μόνο σύνολο καλωδιώσεων μπορούν να χρησιμοποιηθούν διάφορες συσκευές.
Το σημαντικότερο μειονέκτημα της χρήσης διαδρόμου είναι ότι αποτελεί bottleneck στην επικοινωνία των συσκευών, καθώς όλες οι συσκευές πρέπει να τον μοιραστούν.

Για να μπορούν πολλές συσκευές να χρησιμοποιούν τον μοναδικό διάδρομο θα πρέπει να υπάρχει ένα πρωτόκολλο διαμοιρασμού του. Το πρωτόκολλο αυτό θα πρέπει να εγγυάται την επικοινωνία μεταξύ δύο μόνο συστημάτων ανά πάσα στιγμή και τον αποκλεισμό των υπολοίπων από τη χρήση του διαδρόμου όση ώρα διαρκεί η μεταφορά αυτή.

Συχνά χρησιμοποιούνται ιεραρχίες διαδρόμων. Δηλαδή δημιουργούνται μικροί διάδρομοι που συνδέουν μεταξύ τους ορισμένες συσκευές ίδιου τύπου και στη συνέχεια οι μικροί διάδρομοι ενώνονται με τον διάδρομο του παραπάνω επιπέδου χρησιμοποιώντας προσαρμογείς διαδρόμων (bus adapters).

1.4.1 Ανταλλαγή δεδομένων μεταξύ συσκευών και μνήμης

Όλες οι I/O συσκευές πρέπει αφενός να μεταφέρουν τα δικά τους δεδομένα στην κεντρική μνήμη όπου θα τα επεξεργαστεί ο επεξεργαστής και αφετέρου να μπορούν να δέχονται δεδομένα που βρίσκονται στην κεντρική μνήμη. Το είδος της συσκευής καθορίζει και τον τρόπο χειρισμού των δεδομένων που λαμβάνει από την μνήμη.
Ένα πολύ σημαντικό ζήτημα είναι η μέθοδος με την οποία επιτυγχάνεται η μεταφορά δεδομένων από και προς την συσκευή. Στην συνέχεια παραθέτουμε τις τρεις μεθόδους που χρησιμοποιούνται στην πράξη.

Polling

Στην μέθοδο του polling, ο επεξεργαστής μεταφέρει ένα byte κάθε φορά από ή προς την συσκευή και αναλαμβάνει ο ίδιος να ελέγξει την πρόοδο της μεταφοράς. Αυτό το πετυχαίνει ελέγχοντας ανά τακτά χρονικά διαστήματα ειδικούς καταχωρητές της συσκευής που σηματοδοτούν την κατάστασή της. Για να μεταφερθεί ένα byte από την μνήμη σε μια συσκευή, θα πρέπει πρώτα ο επεξεργαστής να φέρει αυτό το byte από την μνήμη σε έναν δικό του καταχωρητή και στη συνέχεια να το μεταφέρει στην συσκευή. Αντίστοιχα για την μεταφορά ενός byte από την συσκευή στην μνήμη, θα πρέπει να μεταφερθεί πρώτα σε έναν καταχωρητή του επεξεργαστή και στη συνέχεια από εκεί στην μνήμη.

Επειδή ο επεξεργαστής ελέγχει ο ίδιος την κατάσταση της συσκευής, υπάρχει σοβαρή πιθανότητα να χρειαστεί να ελέγξει πάρα πολλές φορές τους αυτούς που δείχνουν αν η συσκευή είναι έτοιμη για την επόμενη μεταφορά, πρωτού η συσκευή όντως υποδείξει ότι είναι μπορεί να πραγματοποιηθεί η επόμενη μεταφορά. Αυτό έχει ως αποτέλεσμα να χάνονται πολύτιμοι κύκλοι ρολογιού του επεξεργαστή που θα μπορούσαν να χρησιμοποιηθούν σε κάποια άλλη εργασία. Άρα η τεχνική του polling είναι αποτελεσματική μόνο στην περίπτωση όπου η συσκευή μπορεί πολύ γρήγορα να κάνει διαδοχικές μεταφορές δεδομένων.
Για παράδειγμα δεν θα ήταν καθόλου συνετό να χρησιμοποιηθεί polling για ένα πληκτρολόγιο, καθώς η συχνότητα με την οποία πληκτρολογούνται δεδομένα από τον άνθρωπο είναι τρομαχτικά μικρότερη από εκείνη που ο επεξεργαστής μπορεί να ελέγξει την κατάσταση του πληκτρολογίου.

Interrupts

Στην μέθοδο των interrupts, ο επεξεργαστής δεν κάνει ο ίδιος την αναγνώριση της κατάστασης της συσκευής μετά από κάθε μεταφορά byte, αλλά ειδοποιείται από την ίδια την συσκευή μέσω του μηχανισμού των interrupts ότι είναι έτοιμη για την επόμενη μεταφορά. Δηλαδή ο επεξεργαστής αφού στείλει τις απαραίτητες εντολές για την μεταφορά ενός byte από ή προς την συσκευή, μπορεί να κάνει άλλες εργασίες μέχρι η συσκευή να τον ειδοποιήσει ότι είναι έτοιμη για την επόμενη μεταφορά. Όταν έρθει η ειδοποίηση, ο επεξεργαστής σταματάει την ενέργεια που έκανε και στέλνει στην συσκευή τις εντολές για την επόμενη μεταφορά byte.

Η μέθοδος αυτή είναι πολύ πιο αποτελεσματική από την μέθοδο του polling, ειδικά όταν η συσκευή δεν μπορεί να εκτελέσει γρήγορα διαδοχικές μεταφορές δεδομένων.

Ο λόγος είναι ότι ο επεξεργαστής δεν χάνει χρόνο μεταξύ δύο μεταφορών, αλλά μπορεί να κάνει άλλες εργασίες.

Πρέπει να τονιστεί ότι η μεταφορά των byte και σε αυτή την περίπτωση γίνεται από των καταχωρητών του επεξεργαστή και των καταχωρητών της συσκευής και όχι μεταξύ των καταχωρητών της συσκευής και της κεντρικής μνήμης. Για να εγγραφούν τα δεδομένα των καταχωρητών της συσκευής στην κεντρική μνήμη, πρέπει αφού φθάσουν στον επεξεργαστή, στη συνέχεια εκείνος να τα εγγράψει στην κεντρική μνήμη.

DMA

Η τελευταία και πιο αποτελεσματική μέθοδος ειδικά για συσκευές αποθήκευσης είναι η μέθοδος DMA (Direct Memory Access). Σε αυτή την μέθοδο, ο επεξεργαστής δίνει εντολή στην συσκευή να μεταφέρει από ή προς μια συγκεκριμένη περιοχή μνήμης μια συγκεκριμένη ποσότητα δεδομένων. Η συσκευή προσπελαύνει από μόνη της την υποδεικνυομένη από τον επεξεργαστή περιοχή της κεντρικής μνήμης χωρίς περαιτέρω εμπλοκή του ιδίου. Αφού η συσκευή ολοκληρώσει την μεταφορά, τότε ειδοποιεί τον επεξεργαστή μέσω ενός interrupt ότι η μεταφορά ολοκληρώθηκε. Η μέθοδος αυτή είναι η καλύτερη όταν γίνεται μεταφορά μεγάλων ποσοτήτων δεδομένων καθώς ο επεξεργαστής δεν εμπλέκεται καθόλου στην μεταφορά και μπορεί να εκτελεί απερίσπαστος άλλες εντολές.

Πρέπει να επισημανθούν τρία επιπλέον θέματα σχετικά με το DMA.

Πρώτον, κατά τη διάρκεια της μεταφοράς δεδομένων μεταξύ της συσκευής που κάνει το DMA και της μνήμης, δεν μπορούν άλλα υποσυστήματα του υπολογιστή να προσπελάσουν την μνήμη, αφού η συσκευή χρησιμοποιεί τον διάδρομο. Αυτό σημαίνει ότι ο επεξεργαστής δεν μπορεί να προσπελάσει την κεντρική μνήμη για όσο διαρκεί η μεταφορά μεταξύ συσκευής και μνήμης, αν και εξακολουθεί να έχει την δυνατότητα να προσπελάσει τις cache του (εφόσον δεν υπάρχει περίπτωσης ασυνέπειας όπως θα δούμε παρακάτω). Η αδυναμία του επεξεργαστή να προσπελάσει την κεντρική μνήμη επιφέρει ένα μικρό αρνητικό αντίκτυπο στις επιδόσεις του υπολογιστή.

Δεύτερον, πρέπει να ληφθεί ειδική μέριμνα για να διατηρείται η συνέπεια (consistency) των δεδομένων της/των cache του επεξεργαστή. Μπορούμε να δούμε ότι τίθεται ζήτημα συνέπειας των δεδομένων αυτών αν σκεφτούμε ορισμένα παραδείγματα όπου ο επεξεργαστής διατηρεί σε μια cache του ορισμένα δεδομένα τα οποία υπάρχουν και στην κεντρική μνήμη και μπορούν να λάβουν μέρος σε μια DMA μεταφορά. Αν ο επεξεργαστής συνεχίσει να διαβάζει δεδομένα από την cache ενώ τα αντίστοιχα δεδομένα στην κεντρική μνήμη ενημερώνονται από μια DMA μεταφορά τότε ο επεξεργαστής διαβάζει παλαιά δεδομένα και υπάρχει πρόβλημα ασυνέπειας της cache. Επίσης αν ο επεξεργαστής γράφει δεδομένα σε μία cache και τα αντίστοιχα δεδομένα της μνήμης μεταφέρονται με DMA σε μια συσκευή, αν δεν ενημερωθούν τα δεδομένα αυτά της μνήμης, τότε η συσκευή διαβάζει παλαιά δεδομένα. Για την αντιμετώπιση των προβλημάτων αυτών υπάρχουν λύσεις hardware και software. Και στις δύο λύσεις ο επεξεργαστής ή το λειτουργικό σύστημα αντίστοιχα αναλαμβάνει να ακυρώνει δεδομένα της cache όταν αντιλαμβάνεται ενημέρωση των αντίστοιχων δεδομένων της κεντρικής μνήμης μέσω DMA ενώ γράφει τα δεδομένα της cache

στην μνήμη όταν αυτά συμμετέχουν σε μια DMA.

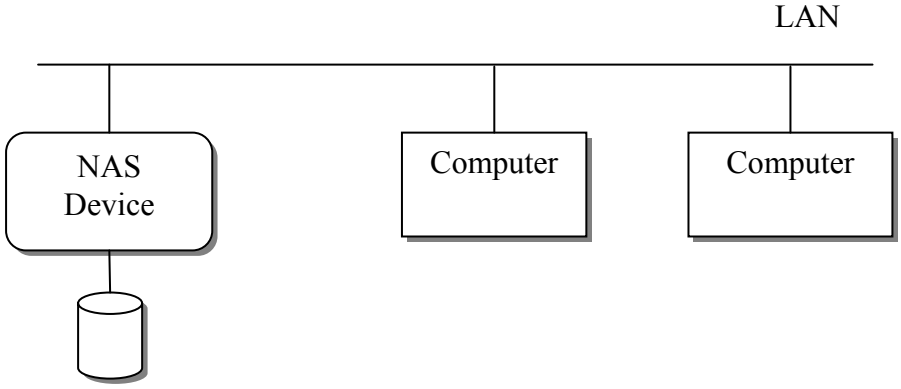
Τρίτον πρέπει να ληφθεί μέριμνα ώστε να προστατεύεται η μνήμη από ακούσιες ή επιτηδευμένες προσβάσεις μέσω DMA, αφού όπως είπαμε με την μέθοδο DMA οι συσκευές που μπορούν να την εφαρμόσουν έχουν ελεύθερη πρόσβαση στην μνήμη. Η προστασία της μνήμης επιτυγχάνεται με την απομόνωση των συσκευών από τις διεργασίες επιπέδου χρήστη. Δηλαδή οι assembly εντολές πρόσβασης στις συσκευές του υπολογιστή μπορούν να εκτελεστούν μόνο από τον πυρήνα και όχι από τις διεργασίες επιπέδου χρήστη. Για την χρησιμοποίηση των συσκευών οι διεργασίες επιπέδου χρήστη πρέπει να χρησιμοποιήσουν τα system calls.

1.5 Τεχνολογίες Δικτυακής Αποθήκευσης

Εκτός από τις τεχνολογίες τοπικής αποθήκευσης, έχουν αναπτυχθεί και τεχνολογίες δικτυακής αποθήκευσης που επιτρέπουν την αποθήκευση και πρόσβαση στα δεδομένα σε συσκευές που είναι προσπελάσιμες μέσω δικτύου.
Οι τεχνολογίες αυτές παρουσιάζονται στη συνέχεια.

1.5.1 NAS (Network Attached Storage)

Μια συσκευή NAS, είναι ένα ειδικού σκοπού αποθηκευτικό σύστημα στο οποίο πολλοί υπολογιστές έχουν πρόσβαση μέσω δικτύου. Οι συσκευές NAS είναι στην ουσία ειδικού σκοπού υπολογιστές (ή ακόμα και κανονικοί υπολογιστές που τρέχουν το απαραίτητο λογισμικό) που έχουν τοπικά συνδεδεμένους σκληρούς δίσκους.
Οι συσκευές αυτές συνδέονται πάνω στο δίκτυο διασύνδεσης και μεταφέρουν τα δεδομένα τους στους άλλους υπολογιστές του δικτύου χρησιμοποιώντας τα πρωτοκόλλα IP και TCP/UDP. Η μεταφορά των δεδομένων γίνεται μόνο στο επίπεδο των αρχείων και ως προγραμματιστικό interface χρησιμοποιείται το RPC (remote procedure call). Όταν δηλαδή ένας υπολογιστής του δικτύου θέλει ένα αρχείο που βρίσκεται στο NAS κάνει την αντίστοιχη κλήση RPC προς το μηχάνημα NAS.
Η υλοποίηση του RPC προγραμματιστικού interface αποτελεί ταυτόχρονα και το σύστημα αρχείων που βλέπουν οι υπολογιστές του δικτύου για τη συσκευή αυτή αποθήκευσης και σε συστήματα UNIX είναι το NFS ενώ στα συστήματα Windows είναι το CIFS.
Για την καλύτερη κατανόηση της τεχνολογίας NAS παρατίθεται η παρακάτω εικόνα.



Σχήμα 1.9: NAS

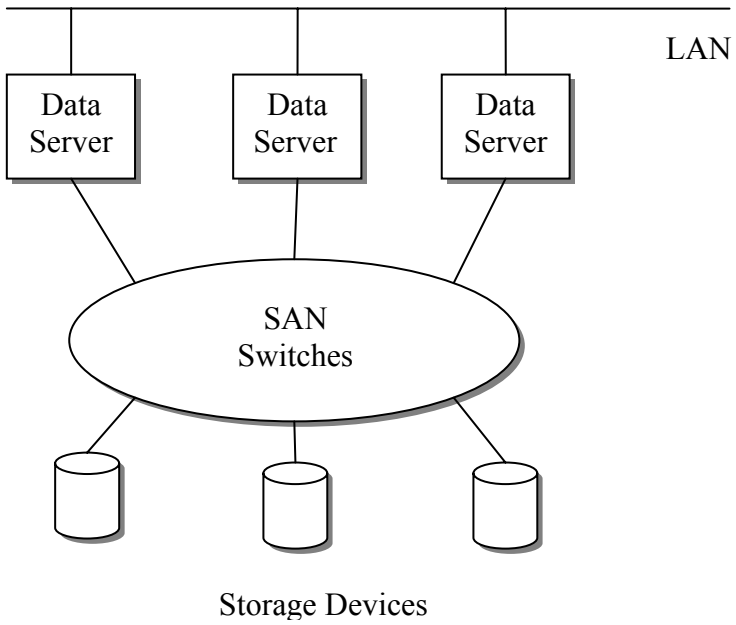
Οι συσκευές NAS παρέχουν έναν απλό και σχετικά φθηνό τρόπο για να έχουν πρόσβαση όλοι οι υπολογιστές ενός δικτύου σε κοινό αποθηκευτικό χώρο. Το τίμημα που πληρώνεται για αυτή την ευκολία είναι οι μειωμένες επιδόσεις σε σχέση με τοπικά συνδεδεμένα μέσα αποθήκευσης.

1.5.2 SAN (Storage Area Network)

Τα SAN (Δίκτυα Αποθηκευτικού Χώρου), είναι ειδικού σκοπού δίκτυα που διασυνδεθούν διαφορών ειδών αποθηκευτικά μέσα, με υπολογιστές που ονομάζονται data servers. Οι data servers μπορούν να κάνουν κοινή χρήση του αποθηκευτικού χώρου, αφού με την βοήθεια ειδικού λογισμικού το λειτουργικό σύστημα κάθε data server αντιλαμβάνεται πως οι αποθηκευτικές συσκευές που περιέχει το SAN είναι τοπικά συνδεδεμένες όπως συμβαίνει για παράδειγμα με έναν δίσκος SCSI.

Το λειτουργικό σύστημα κάθε data server που είναι συνδεδεμένο στο SAN μπορεί να επικοινωνήσει (χρησιμοποιώντας εντολές SCSI συνήθως) με κάθε δίσκο (ανάλογα βεβαία και με τον τρόπο που έχει στηθεί το SAN) και να στέλνει αιτήσεις σε επίπεδο block του δίσκου, όπως ακριβώς γίνεται και με τους πραγματικά τοπικά συνδεδεμένους δίσκους.

Για την καλύτερη κατανόηση της δομής ενός SAN παρατίθεται η παρακάτω εικόνα.



Σχήμα 1.10: SAN

Για την επικοινωνία των αποθηκευτικών μέσων με τους data servers απαιτούνται ειδικά πρωτοκολλά, διασυνδέσεις και συσκευές. Η πιο κοινά χρησιμοποιουμένη τεχνολογία διασύνδεσης είναι αυτή του Fibre Channel (FC). Η τεχνολογία αυτή επιτρέπει την μεταφορά δεδομένων από τους δίσκους του SAN στους data servers, με πολύ μικρό latency και τεράστιο bandwidth, αλλά το κόστος της είναι πολύ υψηλό.

Στην ουσία η τεχνολογία FC μεταφέρει εντολές SCSI από τους data servers στους δίσκους του SAN.

Σε πολλές περιπτώσεις υπάρχει ένα ενδιάμεσο επίπεδο λογισμικού ανάμεσα στο λειτουργικό σύστημα και τους δίσκους του SAN, το οποίο αναλαμβάνει να δημιουργήσει έναν λογικό χώρο αποθήκευσης ορατό από το λειτουργικό σύστημα, το οποίο κρύβει πλήρως την διάταξη του φυσικού χώρου αποθήκευσης. Το λογισμικό αυτό μπορεί για παράδειγμα να δημιουργήσει έναν και μοναδικό λογικό αποθηκευτικό χώρο, έτσι ώστε το λειτουργικό σύστημα να θεωρεί ότι έχει μονό έναν δίσκο συνδεδεμένο, ενώ στη πραγματικότητα ο φυσικός χώρος αποθήκευσης αποτελείται από πολλούς δίσκους.

Στη συνέχεια παρατίθενται οι κυριότερες διαφορές των τεχνολογιών NAS και SAN.

1.5.3 Διαφορές SAN-NAS

Υποδομή

Για την υλοποίηση ενός δικτυού SAN απαιτούνται εξειδικευμένες συσκευές και καλώδια (πέραν των μέσων αποθήκευσης και των υπολογιστών) σε αντίθεση με την τεχνολογία NAS οπού ένα μηχάνημα NAS μπορεί να αποτελείται μονό από ένα κουτί που περιέχει απλούς σκληρούς δίσκους, μια τυπική αρχιτεκτονική pc και ένα ειδικό (περιορισμένων λειτουργιών και δυνατοτήτων) λειτουργικό σύστημα. Η σύνδεση με το υπάρχον τοπικό δίκτυο μπορεί να γίνει όπως ακριβώς στην περίπτωση των υπολογιστών του δικτυού. Όπως γίνεται εύκολα αντιληπτό, το κόστος της υποδομής του SAN είναι αρκετά υψηλότερο από του NAS.

Στην ουσία για την αξιοποίηση της τεχνολογίας SAN σε ένα υπολογιστικό περιβάλλον, είναι απαραίτητη η ύπαρξη δύο δικτύων. Το πρώτο δίκτυο είναι το γνωστό απλό (ή όχι αν χρησιμοποιείται τεχνολογία Myrinet ή Infiniband) δίκτυο που διασυνδέει τους υπολογιστές και το δεύτερο είναι το δίκτυο που διασυνδέει μεταξύ τους αποθηκευτικά μέσα και τελικά τα μέσα αυτά με τους υπολογιστές τους απλού δικτύου. (Σημειώνεται ότι υπάρχουν ερευνητικές προσπάθειες σε εξέλιξη για την εξάλειψη του ειδικού δικτύου διασύνδεσης των αποθηκευτικών μέσων. Βλ "Efficient Block Device Sharing over Myrinet with Memory Bypass," και "Synchronized Send Operations for Efficient Streaming Block I/O over Myrinet,"). Το ειδικό δίκτυο από τη μία πετυχαίνει υψηλότατες ταχύτητες αλλά από την άλλη είναι αρκετά ακριβό, σχετικά δύσκολο στην διαχείριση και υπάρχουν ασυμβατότητες μεταξύ συσκευών διαφόρων κατασκευαστών.

Η τεχνολογία NAS απαιτεί μόνο την ύπαρξη του απλού δικτύου διασύνδεσης. Η συσκευή NAS τρέχει τα γνωστά και απλά στη διαχείριση πρωτόκολλα TCP/IP. Λόγω του commodity υλικού από το οποίο αποτελείται μια συσκευή NAS και των πρωτοκόλλων που τρέχει, είναι αρκετά φθηνή αλλά οι δικτυακές τις επιδόσεις περιορίζονται από τα πρωτόκολλα TCP/IP. (Ο περιορισμός των επιδόσεων σε επίπεδο συστήματος αρχείων θα εξεταστεί παρακάτω).

Μοντέλο Αποθήκευσης

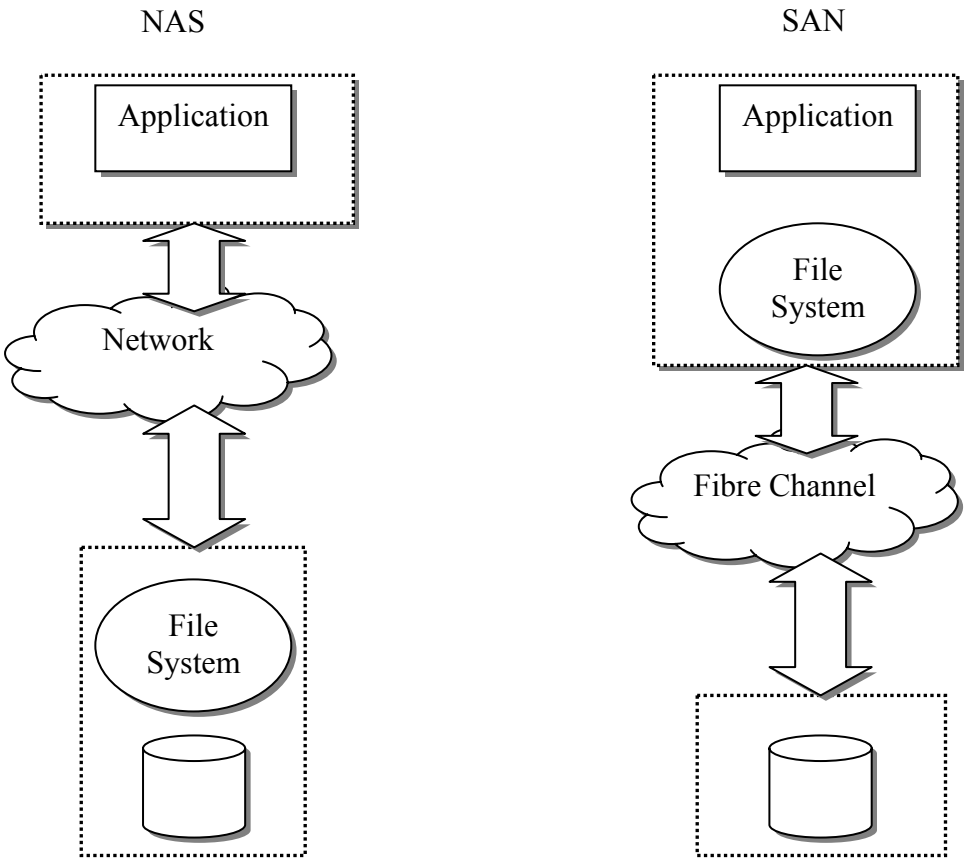
Οι δυο τεχνολογίες διαφοροποιούνται ξεκάθαρα στο μοντέλο αποθήκευσης που χρησιμοποιούν. Στην τεχνολογία NAS, το μηχάνημα NAS τρέχει από μονό του κάποιο σύστημα αρχείων (συνήθως ένα από αυτά που αναφερθήκαν νωρίτερα). Αυτό έχει ως αποτέλεσμα, πρώτον όλες οι μεταφορές που κάνει το μηχάνημα NAS να γίνονται στο επίπεδο του συστήματος αρχείων (δηλαδή μεταφορά αρχείων και metadata για τα αρχεία) και δεύτερον, ότι ο διαμοιρασμός των αρχείων μεταξύ πολλών υπολογιστών που πιθανόν να βρίσκονται στο ίδιο δίκτυο είναι απλός και ελέγχεται από το μηχάνημα NAS (αφού εκεί βρίσκεται και ελέγχεται το σύστημα αρχείων).
Ο διαμοιρασμός είναι απλός αφού όλες οι προσβάσεις στο σύστημα αρχείων περνάνε από το μηχάνημα NAS και συνεπώς σειριοποιούνται πάνω σε αυτό.

Αντίθετα στην τεχνολογία SAN, οι υπολογιστές που συνδέονται στο δίκτυο SAN είναι αποκλειστικά υπεύθυνοι για την δημιουργία του συστήματος αρχείων και οι δίσκοι απλά μεταφέρουν τα δεδομένα τους σε επίπεδο block όταν αυτά τους ζητηθούν. Το μοντέλο αυτό αποθήκευσης είναι ίδιο με το μοντέλο αποθήκευσης που χρησιμοποιείται σε απλούς υπολογιστές οπού οι δίσκοι είναι τοπικά συνδεδεμένοι. Η μονή διαφορά είναι ότι οι δίσκοι δεν βρίσκονται μέσα στο κουτί του υπολογιστή, αλλά σε ένα δικό τους δίκτυο, το δίκτυο SAN.
Σε αντίθεση με την τεχνολογία NAS οπού ο διαμοιρασμός αρχείων είναι σχετικά απλή υπόθεση καθώς το σύστημα αρχείων βρίσκεται στο μηχάνημα NAS και όλες οι αιτήσεις που φθάνουν σε αυτό είναι αιτήσεις στο επίπεδό του, στην τεχνολογία SAN η κοινή χρησιμοποίηση αρχείων από διαφόρους υπολογιστές που πιθανόν να συνδέονται στο SAN δεν είναι καθόλου απλή υπόθεση. Αυτό διότι δεν υπάρχει κεντρικοποιημένο σύστημα αρχείων που να ελέγχει την πρόσβαση στους δίσκους όπως στο NAS και η επικοινωνία με τους δίσκους γίνεται σε επίπεδο block. Αν λοιπόν γινόταν απόπειρα να χρησιμοποιηθεί ένα κοινό σύστημα αρχείων όπως το ext3 πάνω σε ένα SAN, η ανυπαρξία επικοινωνίας μεταξύ των υλοποιήσεων των τμημάτων κώδικα που θα έτρεχαν σε κάθε υπολογιστή και υλοποιούσαν το σύστημα αρχείων, θα το κατάστρεφε. Δύο είναι οι λόγοι που θα επέφεραν αυτή την καταστροφή και αναφέρονται αμέσως παρακάτω:

- Κάθε υπολογιστής θα κράταγε στην κεντρική μνήμη του metadata του συστήματος αρχείων (δηλαδή θα κάνει cache αυτά τα metadata) δημιουργώντας έτσι ασυνέπειες λόγω της ανυπαρξίας συντονισμού των cache των υπολογιστών.
- Δεν θα γινόταν συντονισμένη διαχείριση του ελεύθερου χώρου με αποτέλεσμα οι υπολογιστές να γράφουν ενδεχομένως τα δεδομένα τους στον ίδιο χώρο, καταστρέφοντας τα.

Για τον σωστό διαμοιρασμό αρχείων, απαιτείται η ύπαρξη ενός συστήματος αρχείων που είναι σχεδιασμένο για να λειτουργεί σε τέτοια περιβάλλοντα. Δύο ειδών συστήματα αρχείων μπορούν να επιτελέσουν αυτή την εργασία, τα Συστήματα Αρχείων Κοινού Μέσου (Shared Disk File Systems) και τα Παράλληλα Συστήματα Αρχείων (Parallel File Systems).
Τα παράλληλα συστήματα αρχείων εκτός από την διασφάλιση της ακεραιότητας του συστήματος αρχείων κατά τον διαμοιρασμό αρχείων σε πολλούς υπολογιστές, μπορούν να αυξήσουν τις I/O επιδόσεις αξιοποιώντας τους πολλαπλούς δίσκους του SAN. Περισσότερες πληροφορίες για τα συστήματα αυτά αρχείων υπάρχουν στο

κεφάλαιο 4.
Στην παρακάτω εικόνα φαίνεται το μοντέλο αποθήκευσης που χρησιμοποιούν οι τεχνολογίες SAN και NAS αντίστοιχα.



Σχήμα 1.11 – Διαφορά μοντέλου αποθήκευσης σε SAN και NAS

Όπως φαίνεται από τα παραπάνω, ένα σοβαρό πλεονέκτημα που έχουν τα μηχανήματα NAS είναι η ευκολία με την οποία μπορούν να μοιράσουν τα αρχεία τους σε όλους τους υπολογιστές που βρίσκονται επάνω στο δίκτυο. Αντίθετα, στην τεχνολογία SAN όταν κανείς θέλει να χρησιμοποιεί τα μέσα αποθήκευσης του SAN σε πολλούς υπολογιστές θα πρέπει να εγκαταστήσει ένα κατάλληλο σύστημα αρχείων στους υπολογιστές που θέλουν να έχουν πρόσβαση στον κοινό αποθηκευτικό χώρο του SAN. Αν για κάποιο λόγο ήθελε κανείς να χρησιμοποιήσει το SAN με έναν μόνο υπολογιστή, θα αρκούσε να χρησιμοποιήσει ειδικό λογισμικό που κάνει τους δίσκους του SAN ορατούς στο λειτουργικό σύστημα του υπολογιστή (ως έναν ή πολλούς) και να στήσει πάνω σε κάθε partition που βλέπει το λειτουργικό, ένα ξεχωριστό απλό σύστημα αρχείων όπως το ext3.

1.5.4 Επιπτώσεις της αρχιτεκτονικής των SAN και NAS στην επεκτασιμότητα και την ταχύτητα

Η αρχιτεκτονική που ακολουθούν οι δύο τεχνολογίες έχει σημαντική επίπτωση στην δυνατότητα επέκτασής τους, δηλαδή πόσοι υπολογιστές και δίσκοι μπορούν να

χρησιμοποιηθούν για ταυτόχρονη προσπέλασή τους, χωρίς σημαντική μείωση των επιδόσεων καθώς και στην μέγιστη ταχύτητα με την οποία οι υπολογιστές του δικτύου έχουν πρόσβαση στον κοινό αποθηκευτικό χώρο.

Ας εξετάσουμε πρώτα την τεχνολογία NAS. Όπως συμβαίνει με κάθε σύστημα που ακολουθεί την αρχιτεκτονική Server-Client όταν υπάρχει ένας και μοναδικός Server, αναγκαστικά αποτελεί bottleneck. Ο Server στην περίπτωσή μας είναι το μηχάνημα NAS και Client οι υπόλοιποι υπολογιστές του δικτύου. Εφόσον υπάρχει μόνος ένας Server, δεν τίθεται ζήτημα επέκτασης (scalling) της αρχιτεκτονικής. Όσο περισσότεροι είναι οι Client, τόσο περισσότερο φορτίο επωμίζεται ο Server και τόσο περισσότερο μειώνεται η ταχύτητα απόκρισής του. Επίσης με την αύξηση του αριθμού των Client, αυξάνεται και το I/O load του δικτύου διασύνδεσης που συντελεί ακόμα περισσότερο στην αύξηση του χρόνου απόκρισης του Server για τα δεδομένα που του ζητούνται. Επίσης σε πολλά μηχανήματα NAS δεν μπορεί να αναβαθμιστεί το hardware, θέτοντας εκ των πραγμάτων ένα άνω φράγμα στις αιτήσεις για δεδομένα που μπορεί να εξυπηρετήσει σε δεδομένο χρόνο.
Στην ουσία το NAS είναι μια φθηνή τεχνολογία που μπορεί να εξυπηρετήσει τις δικτυακές ανάγκες αποθήκευσης λίγων κόμβων όταν δεν υπάρχει μεγάλη απαίτηση για γρήγορη εξυπηρέτηση των αιτήσεων.

As δούμε τώρα την τεχνολογία SAN. Σε αυτή την τεχνολογία, το δίκτυο SAN ως υλικό υπόβαθρο δίνει την δυνατότητα

- για επίτευξη μεγάλων ρυθμών διαμεταγωγής δεδομένων από και προς τους δίσκους του δικτύου
- για scalling σε μεγάλο αριθμό δίσκων και υπολογιστών

Αλλά όπως έχουμε αναφέρει το δίκτυο SAN δεν μπορεί να χρησιμοποιηθεί από μόνο του. Είναι απαραίτητη η ύπαρξη ενός κατάλληλου συστήματος αρχείων για να είναι δυνατός ο διαμοιρασμός αρχείων μεταξύ των υπολογιστών που συνδέονται στο SAN. Η αρχιτεκτονική σχεδίαση καθώς και ορισμένες λεπτομέρειες σχεδίασης παίζουν πολύ μεγάλο ρόλο στο κατά πόσον μπορεί να επεκταθεί ένα cluster (δηλαδή ένα σύνολο από υπολογιστές που συνδέονται στο δίκτυο διασύνδεσης και τρέχουν το ίδιο λογισμικό) που χρησιμοποιεί ένα παράλληλο ή ένα σύστημα αρχείων κοινού μέσου.
Οι σημαντικές αρχιτεκτονικές αποφάσεις που επηρεάζουν την δυνατότητα επέκτασης είναι η ύπαρξη ή όχι ρόλων Server μεταξύ των κόμβων, η δυνατότητα ανάκαμψης από σφάλματα (recovery) κ.α. Οι λεπτομέρειες υλοποίησης που είναι σημαντικές αφορούν το πρωτόκολλο επικοινωνίας των κόμβων, την διαχείριση κλειδωμάτων για τον συντονισμό των client, θέματα caching κτλ.

Η ταχύτητα με την οποία θα εξυπηρετούνται οι αιτήσεις για πρόσβαση στον δίσκο επηρεάζεται κυρίως από την δυνατότητα ή όχι αποφυγής των συγκρούσεων (conficts) για την απόκτηση κλειδωμάτων, η δυνατότητα παράλληλης πρόσβασης σε πολλούς δίσκους ταυτόχρονα, η δυνατότητα πολλών κόμβων να προσπελαύνουν ταυτόχρονα διαφορετικά μέρη ενός αρχείου καθώς και η αποτελεσματική χρήση της cache.

Για να καταδείξουμε πόσο σημαντικό είναι το Σύστημα Αρχείων Κοινού Μέσου για το scalling του cluster, αναφέρουμε πως οι αρχικές υλοποιήσεις του GFS δεν επέτρεπαν πρακτικά πάνω από 4 κόμβους να μοιράζονται το σύστημα αρχείων, σε αντίθεση με τις τελευταίες εκδόσεις του GPFS που μπορούν να χρησιμοποιηθούν αποτελεσματικά σε υπολογιστικά περιβάλλοντα με αριθμό μεγαλύτερο των 2000

κόμβων και είναι εγκατεστημένα σε πολλούς από τους πιο δυνατούς υπερυπολογιστές του πλανήτη.

Ένα αντίστοιχο παράδειγμα από την περιοχή των Δικτύων Υπολογιστών που δείχνει την επίδραση της αλληλεπίδραση δύο συστημάτων όσον αφορά το scalling είναι το εξής: Το υλικό μέσο μεταφοράς της πληροφορίας στο Ethernet (δηλαδή το καλώδιο) δίνει από μόνο του την δυνατότητα για επίτευξη μεγάλων ρυθμών διαμεταγωγής δεδομένων σε πολλούς υπολογιστές. Θα πρέπει όμως να υπάρχει και ένα κατάλληλο πρωτόκολλο επίλυσης συγκρούσεων και μεταφοράς δεδομένων για να εκμεταλλευτεί αυτή την υποδομή. Αν το πρωτόκολλο είναι καλό, τότε μπορούν πολλοί υπολογιστές να ανταλλάσουν δεδομένα με μεγάλους ρυθμούς διαμεταγωγής, ενώ αν δεν είναι σωστά σχεδιασμένο τότε διαμεταγωγή είναι πολύ μικρότερη από την θεωρητικά μέγιστη. Στην περίπτωση του SAN και του συστήματος αρχείων κοινού μέσου, αν το δεύτερο είναι καλό, τότε έχουμε μεταφορά δεδομένων σε μεγάλες ταχύτητες για πολλούς κόμβους. Αν όμως δεν είναι προσεκτικά σχεδιασμένο, το cluster δεν θα είναι επεκτάσιμο σε πολλούς κόμβους.

1.5.5 Θέματα Caching σε SAN και NAS

Όπως και σε όλα τα υπολογιστικά συστήματα, η αποδοτική χρήση της cache μπορεί να επιφέρει σημαντική αύξηση της ταχύτητας του συγκεκριμένου συστήματος, αλλά η αποδοτικότερη δυνατή χρήση της είναι ένα δύσκολο και κάθε άλλο παρά τετριμμένο, ζήτημα.

Στην περίπτωση της τεχνολογίας NAS που έχουμε διακριτούς ρόλους Client και Server όσον αφορά την πρόσβαση στα δεδομένα, πρέπει να διακρίνουμε ανάμεσα στο Server Side Caching και στο Client Side Caching. Η περίπτωση του Server Side Caching είναι ακριβώς ίδια με την περίπτωση να χρησιμοποιείται ένα από σύστημα αρχείων για πρόσβαση σε τοπικά δεδομένα. Κάθε αίτηση που γίνεται για πρόσβαση στον δίσκο του Server έχει ως αποτέλεσμα να μπαίνουν στην cache τα αντίστοιχα δεδομένα. Με αυτό τον τρόπο όταν ένας Client ζητά τα συγκεκριμένα δεδομένα από τον Server, αυτός μπορεί να αποφύγει την πρόσβαση στους δίσκους του και να στείλει τα δεδομένα που βρίσκονται στην cache.
Η περίπτωση του Client Side Caching αφορά την δυνατότητα των Client να βάζουν στην cache τους δεδομένα που ζήτησαν από τον Server προκειμένου να αποφύγουν την αίτηση προς τον Server σε περίπτωση που χρειαστούν στο μέλλον τα ίδια δεδομένα. Έτσι κάθε φορά που ένας Client χρειάζεται δεδομένα που βρίσκονται στον Server, ελέγχει πρώτα αν υπάρχουν στην δική του cache και αν όχι τα ζητά από τον Server. Επειδή οι κόμβοι που ζητούν δεδομένα από τον Server δεν συντονίζονται μεταξύ τους (αφού δεν γνωρίζουν καν την ύπαρξη των άλλων Client), για να διατηρηθεί η συνέπεια των cache τους (cache consistency), ο Server πρέπει να παίξει ενεργό ρόλο διατηρώντας πληροφορίες κατάστασης για τις cache αυτές. Ανάλογα με το πρωτόκολλο διατήρησης της συνέπειας που χρησιμοποιείται, ο Server μπορεί να ειδοποιεί τον εκάστοτε κόμβο ότι τα δεδομένα της cache του δεν είναι συνεπή, ή να απαντάει σε ερωτήσεις που κάνουν οι Client για την συνέπεια των cache τους.
Βεβαίως κάθε σύστημα αρχείων που χρησιμοποιείται στην τεχνολογία NAS, χειρίζεται το caching, αλλά ακολουθεί τις γενικές αρχές που παρουσιάστηκαν παραπάνω.

Η χρησιμοποίηση της cache στην τεχνολογία SAN εξαρτάται ολοκληρωτικά από το σύστημα αρχείων που εγκαθίσταται στους υπολογιστές που συνδέονται στο δίκτυο SAN και σχηματίζουν το cluster. Κάθε τέτοιο σύστημα αρχείων χρησιμοποιεί με τον δικό του τρόπο το caching ανάλογα με την αρχιτεκτονική του και τις λεπτομέρειες υλοποίησής του.

Όταν πάνω στο SAN έχει στηθεί ένα σύστημα αρχείων κοινού μέσου, τότε όλοι οι κόμβοι θεωρούνται ομότιμοι όσον αφορά την πρόσβαση στα δεδομένα και συνήθως κάθε κόμβος του cluster έχει μια δική του cache (που αποτελεί κομμάτι της κεντρικής μνήμης που διαχειρίζεται ο πυρήνας του λειτουργικού συστήματος) όπου αποθηκεύονται δεδομένα που έχει χρησιμοποιήσει πρόσφατα.

Τα δεδομένα αυτά μπορούν να χρησιμοποιούνται στη συνέχεια άμεσα από τον ίδιο τον κόμβο όταν δεν παραβιάζεται η συνέπεια του συστήματος αρχείων. Ένα πολύ σημαντικό στοιχείο είναι η αλληλεπίδραση της διαχείρισης κλειδωμάτων και της διαχείρισης της cache.

Όταν ένας κόμβος διαθέτει συγκεκριμένο κλείδωμα για ένα πλήθος δεδομένων μπορεί να χρησιμοποιήσει την cache με τον αντίστοιχο τρόπο. Όταν ανακαλείται το κλείδωμα, τότε ο κόμβος ακυρώνει τα αντίστοιχα δεδομένα της cache του και αν αυτά έχουν τροποποιηθεί, θα πρέπει να γραφτούν στον αποθηκευτικό χώρο. Ο τρόπος απόκτησης και αποδέσμευσης των κλειδωμάτων και το μέγεθος του granularity που παρέχεται από το σύστημα αρχείων καθορίζουν σε μεγάλο βαθμό την αποδοτικότητα του caching.

Για παράδειγμα έχει μεγάλη διαφορά αν ένα σύστημα αρχείων δίνει την δυνατότητα σε κόμβους που έχουν διαβάσει και cache-αρει ένα μεγάλο αρχείο που στη συνέχεια τροποποιείται από κάποιον κόμβο, να διαβάσουν μόνο το τροποποιημένο μέρος του αρχείου, αντί για ολόκληρο το αρχείο από την αρχή. Η δυνατότητα αυτή εξαρτάται από το granularity των κλειδωμάτων και αν αυτό μπορεί να τροποποιηθεί δυναμικά ανάλογα με την περίπτωση.

Στην περίπτωση που πάνω στο SAN έχει στηθεί ένα παράλληλο σύστημα αρχείων, τότε όπως θα δούμε αργότερα υπάρχουν ρόλοι Client και Server και το παράλληλο σύστημα αρχείων καθορίζει τις πολιτικές του Client και του Server Side Caching.

1.5.6 Τεχνολογία Fibre Channel

Το Fibre Channel είναι μια τεχνολογία δικτύωσης που χρησιμοποιείται κυρίως για την δικτύωση συσκευών αποθήκευσης αλλά και δευτερευόντως για την δικτύωση υπολογιστικών συστημάτων. Η τεχνολογία αυτή είναι η κύρια τεχνολογία που χρησιμοποιείται για την δημιουργία SAN και αν και αρχικά χρησιμοποιήθηκε σε περιβάλλοντα υπερυπολογιστών, πλέον έχει περάσει και σε εταιρικά και ερευνητικά περιβάλλοντα.

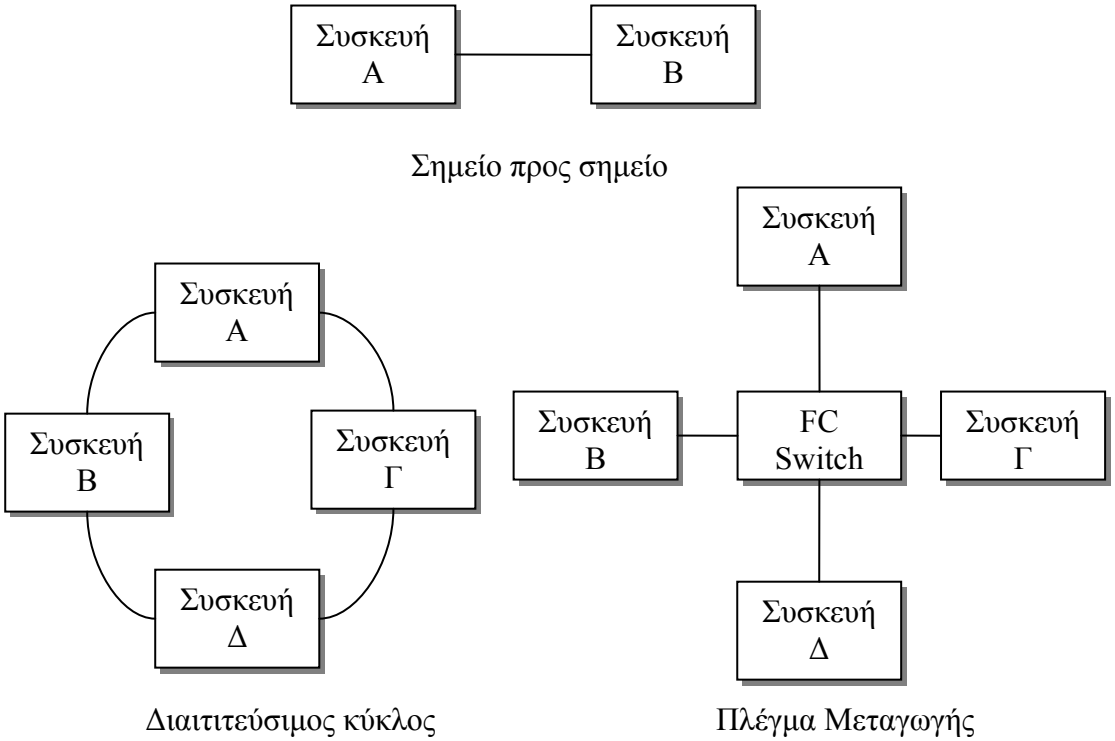
Μπορεί κανείς να διακρίνει τρεις διαφορετικές τοπολογίες σύνδεσης που χρησιμοποιούνται στην τεχνολογία Fibre Channel. Οι συνδέσεις πραγματοποιούνται μεταξύ οντοτήτων που επικοινωνούν μέσω του δικτύου. Οι τρεις τοπολογίες είναι:

- Σημείο προς Σημείο (Point to Point). Σε αυτή την συνδεσμολογία δύο συσκευές συνδέονται μεταξύ τους.
- Διαιτιτεύσιμος Κύκλος (Arbitrary Loop). Σε αυτή την τοπολογία όλες οι συσκευές συνδέονται μεταξύ τους σε έναν δακτύλιο και μπορούν να επικοινωνούν μεταξύ τους μέσω αυτού. Κάθε στιγμή μπορούν να επικοινωνούν μόνο δύο συσκευές με τις άλλες να αναμένουν την

ολοκλήρωση αυτής της επικοινωνίας. Εναλλακτικά ο διαιτιτεύσιμος κύκλος μπορεί να σχηματιστεί συνδέοντας όλες τις συσκευές σε ένα κατάλληλο hub.

- Πλέγμα Μεταγωγής (Switched Fabric). Σε αυτή την τοπολογία όλες οι συσκευές συνδέονται πάνω σε ένα Fibre Channel μεταγωγέα. Οι συσκευές που συνδέονται πάνω στο μεγαγωγέας μπορούν να επικοινωνήσουν ταυτόχρονα μεταξύ τους σαν να είχαν όλοι με όλους μια σύνδεση σημείο προς σημείο. Η τοπολογία αυτή προσφέρει την μέγιστη απόδοση από τις τρεις, αλλά επίσης έχει και το υψηλότερο κόστος.

Για την καλύτερη κατανόηση των τοπολογιών αυτών παρατίθεται η παρακάτω εικόνα.



Σχήμα 1.12: Τοπολογίες Fibre Channel

Το φυσικό μέσο που χρησιμοποιείται για τις συνδέσεις μπορεί να είναι οπτικές είναι ή καλώδια χαλκού.

Όπως και πολλές άλλες τεχνολογίες δικτύωσης, το Fibre Channel είναι μια τεχνολογία που χωρίζεται σε διακριτά επίπεδα. Τα επίπεδα είναι:

1. FC0. Αποτελεί το φυσικό επίπεδο.
2. FC1. Το επίπεδο μεταφοράς.
3. FC2. Το επίπεδο δικτύου
4. FC3. Ένα λεπτό επίπεδο σχεδιασμένο να παρέχει πρόσθετες υπηρεσίες, όπως για παράδειγμα η κρυπτογράφηση
5. FC4. Το επίπεδο όπου γίνεται η ενσωμάτωση των πρωτοκόλλων τα οποία είναι προορισμένα να μεταφέρει το Fibre Channel. Για παράδειγμα αυτό το επίπεδο χρησιμοποιείται για την ενσωμάτωση των εντολών SCSI που

προορίζονται για τα αποθηκευτικά μέσα και την ενσωμάτωση των IP πακέτων σε περιπτώσεις χρησιμοποίησης του Fibre Channel για δικτύωση υπολογιστών.

Στον παρακάτω πίνακα παρέχεται μια σύγκριση μεταξύ του επιπέδου ζεύξης των τεχνολογιών Fibre Channel, Gigabit Ethernet και ATM.

	Fibre Channel	**Gigabit Ethernet**	**ATM**
Εφαρμογές	Δίκτυα αποθήκευσης (SAN), δίκτυα διασύνδεσης, δίκτυα διανομής πολυμεσικού περιεχομένου	Δίκτυα διασύνδεσης	Δίκτυα διασύνδεσης, δίκτυα διανομής πολυμεσικού περιεχομένου
Ρυθμός διαμεταγωγής	1.06Gbps	1.25Gbps	622Mbps
Επεκτασιμότητα σε υψηλότερους ρυθμούς διαμεταγωγής	2.12Gbps, 4.24Gbps	Ακαθόριστο	1.24Gbps
Ύπαρξη υπηρεσία αξιόπιστης μεταφοράς δεδομένων	Ναι	Όχι	Όχι
Απώλεια δεδομένων λόγω υψηλού φορτίου	Όχι	Ναι	Ναι
Μέγεθος πλαισίου	Μεταβλητό, από 0 έως 2KB	Μεταβλητό, από 0 έως 1.5KB	Σταθερό, 53KB
Είδος καλωδιώσεων	Χάλκινα και οπτικών ινών	Χάλκινα και οπτικών ινών	Χάλκινα και οπτικών ινών
Υποστηριζόμενα πρωτόκολλα ανώτερου επιπέδου	TCP/IP, SCSI, Video	TCP/IP	TCP/IP, Video

Πίνακας 1.1: Σύγκριση Fibre Channel, Gigabit Ethernet, ATM

Κεφάλαιο 2.
Γενικά Χαρακτηριστικά Αρχείων και Συστημάτων Αρχείων

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα περιγραφούν τα γενικά χαρακτηριστικά που έχουν τα αρχεία και τα συστήματα αρχείων. Επίσης θα υπάρξει μια γενική ανάλυση του caching των αρχείων.

2.2 Αρχεία (Files)

Κάθε λειτουργικό σύστημα προκειμένου να κάνει διευκολύνει την χρήση του υπολογιστή από τους χρήστες, εισάγει την έννοια του αρχείου και αντιστοιχεί τα δεδομένα των αρχεία στον πραγματικό χώρο αποθήκευσης.
Ουσιαστικά το αρχείο είναι μια επώνυμη συλλογή από σχετιζόμενα δεδομένα που αποθηκεύονται σε αποθηκευτικά μέσα. Από την σκοπιά του χρήστη, το αρχείο είναι η μικρότερη μονάδα που έχει στη διάθεσή του για ν' αποθηκεύσει δεδομένα. Από τη σκοπιά του λειτουργικού συστήματος, το αρχείο είναι μια συλλογή δεδομένων. Η μικρότερη δυνατή μονάδα αποθήκευσης δεδομένων ενός αρχείου είναι το block δεδομένων. Δηλαδή κάθε αρχείο μη μηδενικού μεγέθους αποτελείται από τουλάχιστον ένα block δεδομένων. Τα blocks δεδομένων μπορούν ν' αποθηκευτούν οπουδήποτε στον δίσκο.
Δεν είναι απαραίτητο δηλαδή για ένα αρχείο να αποθηκεύεται σε συνεχόμενα blocks του δίσκου. Μπορούν ομάδες από blocks να αποθηκευτούν σε διάφορα σημεία του δίσκου, προκαλώντας έτσι κατακερματισμό του αρχείου. Ο κατακερματισμός του αρχείου αυξάνει το χρόνο πρόσβασης στα δεδομένα του αρχείου στην περίπτωση που αυτά διαβάζονται σειριακά, καθώς εισάγει επιπλέον χρόνους αναζήτησης του δίσκου.

Τα αρχεία συνήθως αναπαριστούν πληροφορίες του χρήστη ή προγράμματα. Τα δεδομένα που περιέχουν μπορεί να είναι σε δυαδική, αλφαριθμητική, αριθμητική ή αλφαβητική μορφή. Επίσης μπορεί να έχουν συγκεκριμένη δομή να είναι τελείως αδόμητα, όπως τα αρχεία XML και τα αρχεία κειμένου αντίστοιχα. Βλέπουμε δηλαδή ότι η έννοια του αρχείου είναι εξαιρετικά γενικευμένη.

Κάθε αρχείο περιέχει ορισμένες ιδιότητες με πιο σημαντική (για τον χρήστη) το όνομά του. Βεβαίως οι ιδιότητες που έχει ένα αρχείο διαφέρουν από λειτουργικό σε λειτουργικό (και ενδεχομένως από σύστημα αρχείων σε σύστημα αρχείων) αλλά γενικά περιλαμβάνουν τις παρακάτω:

Όνομα

Το όνομα του αρχείου υπάρχει μόνο για διευκόλυνση του χρήστη. Είναι επίσης η μόνη ιδιότητα που κρατείται σε αναγνώσιμη από τον χρήστη μορφή.

Αναγνωριστικό

Κάθε αρχείο έχει ένα μοναδικό αναγνωριστικό, συνήθως έναν αριθμό, προκειμένου να μπορεί να αναγνωριστεί μοναδικά μέσα στο σύστημα αρχείων στο οποίο ανήκει.

Τύπος

Σε ορισμένα λειτουργικά συστήματα που υποστηρίζουν διαφορετικούς τύπους αρχείων, αποθηκεύεται και ο τύπος αυτός του αρχείου.

Τοποθεσία

Αυτή η πληροφορία δείχνει την συσκευή στην οποία είναι αποθηκευμένο το αρχείο καθώς και την ακριβή τοποθεσία (ή τις ακριβείς τοποθεσίες αν το αρχείο είναι κατακερματισμένο) αποθήκευσης μέσα στην συσκευή.

Μέγεθος

Αποθηκεύεται το τρέχων μέγεθος του αρχείου και πιθανόν το μέγιστο μέγεθος που μπορεί να πάρει.

Προστασία

Εδώ αποθηκεύονται δεδομένα που καθορίζουν ποιοι χρήστες έχουν την δυνατότητα να διαβάζουν, να γράφουν και να εκτελούν το αρχείο.

Χρόνος

Αποθηκεύονται πληροφορίες που έχουν να κάνουν με τον χρόνο δημιουργίας, τελευταίας πρόσβασης και τελευταίας τροποποίησης του αρχείου.

Όλες αυτές οι πληροφορίες για τα αρχεία αποθηκεύονται στα μόνιμα μέσα αποθήκευσης σε δομές δεδομένων που καθορίζει και διατηρεί το σύστημα αρχείων. Στα συστήματα αρχείων που υποστηρίζονται από τα λειτουργικά συστήματα τύπου UNIX, για κάθε αρχείο υπάρχουν δύο δομές δεδομένων που τα αναπαριστούν. Η πρώτη ονομάζεται dentry (directory entry) και αντιστοιχεί το όνομα του αρχείο με το αναγνωριστικό του. Η δεύτερη δομή ονομάζεται inode που περιέχει όλες τις ιδιότητες του αρχείου πλην του ονόματος καθώς και επιπλέον πληροφορίες που διατηρεί το σύστημα αρχείων. Οι inode δομές μπορεί να καταλαμβάνουν αρκετό αποθηκευτικό χώρο (ακόμα και 1KB η κάθε μία) και για την γρήγορη προσπέλασή τους, μπαίνουν σε ειδική cache που κρατά το λειτουργικό σύστημα στην κεντρική μνήμη του υπολογιστή.

Σε κάθε αρχείο μπορούν να γίνουν ορισμένες ενέργειες που αφορούν την διαχείρισή

τους. Από την σκοπιά του προγραμματιστή εφαρμογών (application programmer), οι λειτουργίες αυτές επιτελούνται με την χρήση των κλήσεων συστήματος (system calls) που προσφέρει το λειτουργικό σύστημα. Οι λειτουργίες που συνήθως επιτρέπονται από το προγραμματιστικό περιβάλλον του λειτουργικού συστήματος (και του συστήματος αρχείων) πάνω στα αρχεία είναι οι εξής:

Δημιουργία

Η δημιουργία των αρχείων περιλαμβάνει δύο στάδια. Την δέσμευση χώρου στον δίσκο για τα δεδομένα του αρχείου και προσθήκη μιας καταχώρησης για το αρχείο μέσα στον κατάλογο του συστήματος αρχείων στον οποίο δημιουργείται.

Ανάγνωση

Η ανάγνωση πραγματοποιείται με μια κλήση συστήματος με παραμέτρους το προς ανάγνωση αρχείο, το πλήθος των προς ανάγνωση δεδομένων και την περιοχή της μνήμης που θέλουμε να μεταφερθούν τα δεδομένα που θα διαβαστούν. Το λειτουργικό σύστημα πρέπει στη συνέχεια να ψάξει στον κατάλογο του συστήματος αρχείων για το συγκεκριμένο αρχείο και να οργανώσει την μεταφορά των δεδομένων από το αποθηκευτικό μέσο στην κεντρική μνήμη. Για να έχει νόημα η ανάγνωση, θα πρέπει το λειτουργικό σύστημα να κρατά και να ενημερώνει σε κάθε ανάγνωση έναν δείκτη μέσα στο αρχείο που δείχνει το σημείο μέσο στο αρχείο από όπου έγινε η τελευταία ανάγνωση.

Εγγραφή

Η εγγραφή σε ένα αρχείο γίνεται με μια κλήση συστήματος με παραμέτρους το προς εγγραφή αρχείο, το πλήθος των προς εγγραφή δεδομένων καθώς και την περιοχή της μνήμης από την οποία θα παρθούν τα δεδομένα που θα γραφτούν στο αποθηκευτικό μέσο. Όπως και στην ανάγνωση, το λειτουργικό θα πρέπει αρχικά να βρει το αρχείο μέσα στον κατάλογο και στη συνέχεια να οργανώσει την μεταφορά των δεδομένων από την μνήμη προς το αποθηκευτικό μέσο. Ο δείκτης που δείχνει το σημείο στο οποίο θα γίνει η εγγραφή είναι συνήθως ο ίδιος με τον δείκτη που χρησιμοποιείται στην περίπτωση της ανάγνωσης.

Μετακίνηση εντός του αρχείου

Η λειτουργία αυτή αφορά την μετακίνηση του δείκτη που δείχνει την τελευταία θέση που προσπελάστηκε στο αρχείο. Η μετακίνηση αυτή γίνεται ώστε να μπορούν οι εφαρμογές να προσπελάσουν στη συνέχεια (με ανάγνωση ή εγγραφή) οποιοδήποτε μέρος του αρχείου παρακάμπτοντας την default σειριακή προσπέλαση. Σημειώνουμε ότι η λειτουργία αυτή δεν υποστηρίζεται σε συστήματα που υποστηρίζουν μόνο σειριακή προσπέλαση των αρχείων.

Διαγραφή

Για να γίνει διαγραφή του αρχείου, το λειτουργικό σύστημα θα πρέπει αφού βρει το αρχείο, να αποδεσμεύει τον αποθηκευτικό χώρο που καταλάμβανε και να διαγράψει τις δομές δεδομένων του συστήματος αρχείων που αναπαριστούν το προς διαγραφή αρχείο.

Μηδενισμός

Αυτή η λειτουργία επιτρέπει στον χρήστη να διαγράφει τα περιεχόμενα του αρχείου χωρίς να σβήσει το ίδιο το αρχείο. Μένουν δηλαδή ανέπαφα τα πεδία των δομών δεδομένων που αναπαριστούν τα αρχείο εκτός από το πεδίο του μεγέθους που μηδενίζεται ενώ αποδεσμεύεται ο χώρος που κατελάμβαναν τα δεδομένα του αρχείου.

Οι λειτουργίες αυτές είναι οι βασικές λειτουργίες που θα πρέπει να επιτρέπει ένα σύστημα αρχείων πάνω στα αρχεία του. Άλλες λειτουργίες που πιθανόν να υποστηρίζονται (όπως μετονομασία του αρχείου, αντιγραφή του κτλ.) μπορούν να υλοποιηθούν ως συνδυασμός των βασικών λειτουργιών. Για παράδειγμα η αντιγραφή ενός αρχείου Α σε ένα αρχείο Β, περιλαμβάνει την δημιουργία του Β, που ακολουθείται από το εγγραφή των δεδομένων που διαβάζονται από το Α, στο Β.

Τα αρχεία όπως είπαμε αποθηκεύουν δεδομένα. Όταν υπάρχει ανάγκη χρησιμοποίησης των δεδομένων αυτών θα πρέπει να χρησιμοποιείται κάποια μέθοδος προσπέλασης των αρχείων. Υπάρχουν διάφορα είδη προσπέλασης που εντάσσονται στις παρακάτω κατηγορίες:

Σειριακή προσπέλαση

Σε αυτό το είδος της προσπέλασης κάθε byte δεδομένων του αρχείου διαβάζεται σειριακά από την αρχή μέχρι το τέλος του. Ο τρόπος αυτός προσπέλασης χρησιμοποιείται από προγράμματα όπως επεξεργαστές κειμένου, μεταγλωττιστές κ.α.

Τυχαία προσπέλαση

Σε αυτό το είδος της προσπέλασης, τα blocks στα οποία αποθηκεύονται τα δεδομένα των αρχείων προσπελάζονται με τυχαία σειρά. Αυτό είναι εφικτό όταν τα blocks από τα οποία αποτελείται το αρχείο αποθηκεύονται σε αποθηκευτικά μέσα που υποστηρίζουν την τυχαία προσπέλαση σε blocks δεδομένων, όπως συμβαίνει με τους σκληρούς δίσκους ή τα οπτικά μέσα αποθήκευσης. Η τυχαία προσπέλαση είναι πολύ χρήσιμη όταν είναι απαραίτητη η άμεση προσπέλαση συγκεκριμένων δεδομένων μέσα σε μεγάλα αρχεία. Τέτοιου είδους πρόσβαση είναι απαραίτητη σε εφαρμογές όπως Συστήματα Διαχείρισης Βάσεων Δεδομένων.

Σύνθετη προσπέλαση

Με βάση την τυχαία προσπέλαση μπορούν να δημιουργούν διαφόρων ειδών σύνθετες προσπελάσεις. Η πιο συνηθισμένη σύνθετη προσπέλαση είναι η προσπέλαση με τη

χρήση ευρετηρίου. Το ευρετήρια είναι δομές δεδομένων (αποθηκευμένες στην αρχή του αρχείου) που περιλαμβάνουν δείκτες προς διάφορα blocks δεδομένων του αρχείου.
Ένα παράδειγμα ευρετηρίου σε αρχείο είναι ένα ταξινομημένο ευρετήριο. Με βάση αυτό μπορούμε να ψάξουμε για συγκεκριμένες πληροφορίες και ακολουθώντας τον αντίστοιχο δείκτη να πάρουμε απευθείας το block που περιλαμβάνει τα σωστά δεδομένα. Όταν συγκρίνουμε την προσπέλαση με και χωρίς ευρετήριο, βλέπουμε ότι σε περιπτώσεις μεγάλων αρχείων, η προσπέλαση με ευρετήριο υπερέχει ξεκάθαρα. Αυτό διότι στην περίπτωση ύπαρξης του ευρετηρίου στην χειρότερη περίπτωση διαβάζονται όλα τα blocks του ευρετηρίου (που σίγουρα θα σχετικά μικρό ποσοστό επί του συνόλου των blocks του αρχείου) συν το ένα block δεδομένων που ψάχνουμε, ενώ όταν δεν υπάρχει ευρετήριο θα πρέπει στην χειρότερη περίπτωση να διαβάσουμε όλα τα blocks του αρχείου.
Σε περιπτώσεις πολύ μεγάλων αρχείων μπορούν να χρησιμοποιηθούν και πολυεπιπέδα ευρετήρια μειώνοντας ακόμα περισσότερο τις προσπελάσεις στον δίσκο.

Ένα πολύ σημαντικό θέμα για την διαχείριση των αρχείων από τους χρήστες είναι η οργάνωσή τους σε καταλόγους.
Τα πρώτα λειτουργικά συστήματα είχαν μόνο έναν κατάλογο και όλα τα αρχεία έπρεπε να αποθηκευτούν εκεί. Αυτό δημιουργούσε προβλήματα αφού με την αύξηση του αριθμού των αρχείων, αφενός μειώνονταν οι ταχύτητες αναζήτησής τους και αφετέρου έπρεπε να εφευρίσκονται συνέχεια νέα ονόματα για τα αρχεία.
Την λύση σε αυτά τα προβλήματα έδωσαν τα ιεραρχικά συστήματα αρχείων. Με αυτά τα αρχεία οργανώνονται σε καταλόγους έτσι ώστε κάθε κατάλογος να περιέχει μια λίστα από αρχεία που σχετίζονται (ή όχι) μεταξύ τους με τρόπο που επιλέγει ο χρήστης. Ξεκινώντας από τον κατάλογο ανώτατου επιπέδου (κατάλογος ρίζας) επιτρέπεται η ύπαρξη καταλόγων και υποκαταλόγων σε οποιοδήποτε βάθος. Με αυτό τον τρόπο η λογική οργάνωση των αρχείων (η φυσική οργάνωσή τους στον δίσκο είναι τελείως διαφορετική) σχηματίζει ένα δένδρο.
Στα ιεραρχικά συστήματα αρχείων, το όνομα του αρχείο δεν αρκεί πλέον για να προσδιοριστεί πλήρως, αλλά απαιτείται η πλήρης διαδρομή από τον κατάλογο ρίζας.

2.3 Συστήματα Αρχείων (File Systems)

Το Σύστημα Αρχείων είναι μια μέθοδος αποθήκευσης και οργάνωσης των αρχείων ενός υπολογιστή προκειμένου να είναι δυνατή η εύκολη ανεύρεση και προσπέλασή τους. Το σύστημα αρχείων είναι το πιο ορατό στους χρήστες, υποσύστημα του λειτουργικού συστήματος.
Το σύστημα αρχείων παρέχει στις εφαρμογές και τους χρήστες μια λογική αφαίρεση του χώρου αποθήκευσης. Σύμφωνα με την αυτή την αφαίρεση ο αποθηκευτικός χώρος αποτελείται από έναν πίνακα από τμήματα σταθερού μεγέθους δεδομένων, τα blocks. Πάνω σε αυτό τον λογικό χώρο μπορούν οι εφαρμογές και οι χρήστες που τις χρησιμοποιούν να αποθηκεύουν τα αρχεία τους.

Το σύστημα αρχείων απαιτεί την επίλυση δύο αρκετά διαφορετικών προβλημάτων.
Το πρώτο πρόβλημα αφορά τον τρόπο με τον οποίο θα είναι ορατό το σύστημα αρχείων στους χρήστες. Περιλαμβάνει τον ορισμό των ιδιοτήτων των αρχείων, των λειτουργιών που μπορούν επιτελεστούν πάνω στα αρχεία και την δομή του καταλόγου.

Το δεύτερο πρόβλημα αφορά την κατασκευή δομών δεδομένων και αλγορίθμων που επιτρέπουν την αντιστοίχηση του λογικού χώρου των αρχείων στο φυσικό μέσο αποθήκευσης.

Ο κώδικας που υλοποιεί ένα σύστημα αρχείων συνήθως οργανώνεται σε λογικά επίπεδα. Τα επίπεδα είναι από πάνω προς τα κάτω τα εξής: λογικό σύστημα αρχείων, οργάνωση αρχείων και το κύριο σύστημα αρχείων.. Κάθε επίπεδο χρησιμοποιεί τις υπηρεσίες του αμέσως παρακάτω επιπέδου και δίνει στο παραπάνω επίπεδο τις νέες υπηρεσίες που υλοποιεί.

Η αρμοδιότητα του κύριου σύστημα αρχείων είναι να δίνει εντολές μεταφοράς blocks δεδομένων προς τους σωστούς οδηγούς συσκευών.
Το επίπεδο οργάνωσης αρχείων γνωρίζει για την ύπαρξη των αρχείων καθώς και για την λογική και φυσική οργάνωσή τους. Κάθε αρχείο καταλαμβάνει έναν αριθμό από N λογικά blocks που αριθμούνται από 0 εώς N-1. Αυτοί οι αριθμοί δεν είναι οι φυσικοί αριθμοί block στους οποίους αποθηκεύονται τα δεδομένα του αρχείου. Η αντιστοίχηση των λογικών και των φυσικών αριθμών block γίνεται σε αυτό το επίπεδο. Επίσης σε αυτό το επίπεδο γίνεται και η διαχείριση του ελεύθερου αποθηκευτικού χώρου.
Στο υψηλότερο επίπεδο, το λογικό σύστημα αρχείων, βρίσκεται η διαχείριση των μεταπληροφοριών (metadata). Τα metadata είναι όλες οι πληροφορίες και οι δομές του συστήματος αρχείων που δεν αποτελούν όμως δεδομένα των αρχείων. Το λογικό σύστημα αρχείων διαχειρίζεται την ιεραρχική δομή του συστήματος αρχείων παρέχοντας στο επίπεδο οργάνωσης αρχείων τις πληροφορίες που χρειάζεται.
Οι οδηγοί συσκευών αν και δεν αποτελούν μέρος του συστήματος αρχείων, εντούτοις αλληλεπιδρούν αρκετά με αυτό και ως εκ τούτου περιγράφεται σύντομα ο ρόλος τους.
Οι οδηγοί των συσκευών αποθήκευσης λοιπόν αναλαμβάνουν να πραγματοποιούν μεταφορές δεδομένων από τις συσκευές προς την μνήμη και αντίστροφα. Βρίσκονται ιεραρχικά κάτω από το κύριο σύστημα αρχείων και λαμβάνουν από αυτό εντολές μεταφοράς συγκεκριμένων blocks της συσκευής αποθήκευσης. Ως έξοδο παράγουν ειδικές εντολές προς τους ελεγκτές των συσκευών για να γίνει η μεταφορά.

Τα συστήματα αρχείων ταξινομούνται συνήθως στα παρακάτω είδη:

Συστήματα αρχείων για χρήση σε τοπικά μέσα αποθήκευσης

Είναι η πιο κοινή περίπτωση συστήματος αρχείων και αναλαμβάνει να εξυπηρετήσει τις ανάγκες εφαρμογών για αποθήκευση σε σκληρούς δίσκους, άλλα αποσπώμενα μαγνητικά αποθηκευτικά μέσα και οπτικά μέσα ή συσκευές flash που είναι τοπικά συνδεδεμένα με τον υπολογιστή.

Δικτυακά Συστήματα Αρχείων

Τα συστήματα αυτά αρχείων χρησιμοποιούνται σε περιβάλλοντα όπου πολλοί υπολογιστές που συνδέονται μέσω δικτύου διασύνδεσης έχουν ανάγκη κοινής χρήσης αρχείων που βρίσκονται σε αποθηκευτικό χώρο που είτε είναι κοινός σε όλους, είτε είναι άμεσα συνδεδεμένος σε ορισμένους μόνο από τους υπολογιστές.

Συστήματα Αρχείων Ειδικού Σκοπού

Η κατηγορία αυτή περιλαμβάνει τα συστήματα αρχείων που δεν εμπίπτουν σε μία από τις δύο παραπάνω κατηγορίες. Ένα τέτοιο παράδειγμα είναι το σύστημα αρχείων procfs ορισμένων UNIX λειτουργικών συστημάτων, που δίνει πληροφορίες σχετικά με την κατάσταση των διεργασιών που τρέχουν.
Ειδικά στην περίπτωση του λειτουργικού συστήματος Linux, το procfs περιλαμβάνει επιπλέον και πληροφορίες που έχουν να κάνουν με την κατάσταση του συστήματος γενικότερα.

2.4 Γενικά για το caching αρχείων

Όλα τα σύγχρονα λειτουργικά συστήματα εκμεταλλεύονται το caching των αρχείων προκειμένου να μειώσουν τους χρόνους ανάγνωσης και εγγραφής των αρχείων. Ο μειωμένος χρόνος προσπέλασης των δεδομένων είναι αποτέλεσμα της τεράστιας διαφοράς που υπάρχει στην ταχύτητα προσπέλασης της μνήμης και του αποθηκευτικού μέσου.
Στη συνέχεια περιγράφουμε το caching των αρχείων που εφαρμόζεται συνήθως από τα λειτουργικά συστήματα για την ανάγνωση και την εγγραφή αρχείων.

Όσον αφορά τις αναγνώσεις, όταν μια διεργασία θέλει να διαβάσει δεδομένα από κάποιο αρχείο, ελέγχεται πρώτα η cache προκειμένου να διαπιστωθεί αν τα ζητούμενα δεδομένα υπάρχουν. Αν τα δεδομένα όντως υπάρχουν στην cache, τότε διαβάζονται κατευθείαν από αυτή χωρίς να χρειάζεται πρόσβαση στον δίσκο. Αν τα δεδομένα δεν υπάρχουν στην cache, τότε αρχικά διαβάζονται από τον δίσκο και γράφονται στην cache ενώ στη συνέχεια αντιγράφονται από την cache στην περιοχή μνήμης της διεργασίας που έκανε την αίτηση ανάγνωσης.

Η cache αξιοποιείται και στις περιπτώσεις της εγγραφής δεδομένων. Όταν μια διεργασία ζητήσει μια εγγραφή στον δίσκο, τα δεδομένα αυτά πρώτα γράφονται στην cache. Η εγγραφή αυτή στην cache πραγματοποιείται είτε τα αντίστοιχα δεδομένα υπήρχαν στην cache (σε παλαιότερη έκδοση) είτε όχι. Ο λόγος που γίνεται η εγγραφή στην cache είναι να εξυπηρετηθούν από την cache μελλοντικές αναγνώσεις των ίδιων δεδομένων.

Στην περίπτωση της εγγραφής πρέπει να επισημανθούν τρία επιπλέον θέματα.
Πρώτον, το system call που κλήθηκε από την εφαρμογή για να πραγματοποιηθεί η εγγραφή, επιστρέφει μόλις τα δεδομένα γραφτούν στην cache και όχι στον δίσκο. Η εφαρμογή **δεν** ενημερώνεται στη συνέχεια με κάποιο ασύγχρονο τρόπο για την τελική εγγραφή των δεδομένων στον δίσκο.
Δεύτερον, τα προς εγγραφή στον δίσκο δεδομένα που βρίσκονται στην cache δεν γράφονται απευθείας στον δίσκο, αλλά η εγγραφή αυτή καθυστερεί. Η καθυστέρηση δημιουργείται προκειμένου να εκμεταλλευτεί η πιθανότητα να υπάρξουν άμεσα, μελλοντικές αιτήσεις εγγραφής των οποίων τα δεδομένα καταλαμβάνουν γειτονικές θέσεις στον χώρο αποθήκευσης με τα αρχικά δεδομένα. Αν γίνει η ταυτόχρονη αυτή εγγραφή μειώνεται το συνολικός χρόνος εγγραφής των δεδομένων αφού εξαλείφεται ο ένας από τους δύο χρόνους αναζήτησης στο αποθηκευτικό μέσο.

Τρίτον, δεν δίνεται καμία εγγύηση από το λειτουργικό σύστημα ότι τα δεδομένα που γράφτηκαν πρώτα στην cache θα γραφτούν και πρώτα στον δίσκο.

Αν και η χρήση της cache στις περισσότερες περιπτώσεις αυξάνει τις επιδόσεις προσπέλασης στον δίσκο και δεν προκαλεί προβλήματα στην διαχείριση του αποθηκευτικού χώρου, εντούτοις υπάρχουν και περιπτώσεις που μπορεί να συμβεί το ακριβώς αντίθετο. Υπάρχουν περιπτώσεις όπου μειώνονται οι επιδόσεις, περιπτώσεις που μπορεί να προκληθεί απώλεια δεδομένων και περιπτώσεις όπου αλλοιώνεται η σημασιολογία της εγγραφής των δεδομένων στο αποθηκευτικό χώρο έτσι όπως την είχαν καθορίσει οι εφαρμογές που τον προσπέλαυναν.
Παραδείγματα των αντίστοιχων περιπτώσεων παρατίθενται αμέσως παρακάτω.

Όταν μια εφαρμογή θέλει να διαβάσει μια μεγάλη ποσότητα δεδομένων από τον δίσκο για την οποία είναι σίγουρη πως δεν θα την ξαναχρειαστεί, η χρήση της cache όχι μόνο δεν μειώνει το χρόνο ανάγνωσης, αλλά τον αυξάνει. Αυτό διότι προσθέτει το επιπλέον κόστος της αντιγραφής των δεδομένων από μια θέση της μνήμης (της θέσης που καταλαμβάνει η cache) σε μια άλλη (της θέσης που αντιστοιχεί στην εφαρμογή) και δεν προσφέρει κανένα πλεονέκτημα για την εφαρμογή, αφού δεν πρόκειται να αναγνωσθούν ξανά τα δεδομένα.

Στις περιπτώσεις της εγγραφής δεδομένων στον δίσκο, επειδή τα δεδομένα γράφονται πρώτα στην cache και μετά με καθυστέρηση στον δίσκο, αν υπάρξει πρόβλημα στο αποθηκευτικό μέσο, πάγωμα του πυρήνα του λειτουργικού, ή κλείσιμο του συστήματος για κάποιο λόγο, τα δεδομένα που βρίσκονται στην cache χάνονται. Αυτό σημαίνει ότι όσες εγγραφές είχαν παραμείνει στην cache και δεν είχαν περάσει στον δίσκο, χάνονται οριστικά.

Μια περίπτωση όπου η διαφορετική σειρά εγγραφής των δεδομένων στην cache από ότι στον δίσκο μπορεί να προκαλέσει πρόβλημα είναι η εξής:
Για τις ανάγκες κάποιας εφαρμογής, χρειάζεται να δημιουργηθεί μια απλά συνδεδεμένη λίστα μέσα σε ένα αρχείο. Η συνετή προγραμματιστική σκέψη είναι δεσμευθεί πρώτα ο χώρος για ένα νέο κόμβο της λίστας και να αποθηκευτεί στο αρχείο και στη συνέχεια να δημιουργηθεί και αποθηκευτεί στο αρχείο ένας δείκτη προς τον νέο κόμβο. Οι ενέργειες αυτές περιλαμβάνουν δύο εγγραφές, μία για τον νέο κόμβο και μία για τον δείκτη και πρέπει να γίνουν με αυτή την σειρά για να αποφύγουμε την περίπτωση του ξεκρέμαστου δείκτη αν ο υπολογιστής παρουσιάσει πρόβλημα μετά την πρώτη εγγραφή.
Άσχετα με το γεγονός ότι η εφαρμογή κάνει τις εγγραφές με αυτή τη σειρά στην cache, αν το λειτουργικό σύστημα αποφασίσει πως θα γίνει πρώτα η εγγραφή του δείκτη αντί του νέου κόμβου, τότε αν ο υπολογιστής παρουσιάσει πρόβλημα μετά την πρώτη εγγραφή, ο νέος δείκτης θα μείνει ξεκρέμαστος. Βλέπουμε δηλαδή ότι η σημασιολογία της εγγραφής αλλοιώθηκε από το λειτουργικό σύστημα παρ' όλο που ο προγραμματιστής έλαβε την κατάλληλη μέριμνα για την αποφυγή αυτού του γεγονότος.
Πρέπει να σημειώσουμε πως αν και οι περιπτώσεις που αναφέραμε είναι δυνατόν να συμβούν, στα σημερινά υπολογιστικά συστήματα η πιθανότητα να συμβούν είναι μικρή.

Για να ξεπεραστούν τα προβλήματα που εισάγει η χρήση της cache καθώς και για άλλους λόγους όπως την δυνατότητα που θέλουν ορισμένες εφαρμογές να δια-

χειρίζονται από μόνες του την cache, τα λειτουργικά συστήματα παρέχουν την δυνατότητα παράκαμψης της cache κατά την ανάγνωση ή εγγραφή αρχείων.

Μέχρι στιγμής όπου αναφέραμε caching αρχείων, υπονοούσαμε το caching των δεδομένων των αρχείων και όχι των δομών δεδομένων του συστήματος αρχείων που αναπαριστούν τα αρχεία. Το λειτουργικό σύστημα όμως κάνει και caching των αυτών των και πιο συγκεκριμένα των dentry δομών και των inode δομών. Κάθε φορά που προσπελάζεται ένα αρχείο, το λειτουργικό σύστημα κρατά στην cache τις δύο αυτές δομές αυξάνοντας έτσι την ταχύτητα προσπέλασης στο αρχείο για μελλοντικές προσπελάσεις του.

Για να ολοκληρώσουμε την παρουσίαση του caching αρχείων αναφέρουμε δύο επιπλέον στοιχεία.
Κάθε λειτουργικό σύστημα έχει διαφορετικές λεπτομέρειες υλοποίησης του caching αρχείων, αλλά γενικά ισχύουν όσα αναφέρθηκαν παραπάνω. Για παράδειγμα μια σημαντική λεπτομέρεια είναι αν το λειτουργικό θα διαχειρίζεται το caching με ξεχωριστούς μηχανισμούς ή θα ενοποιείται στη διαχείριση της εικονικής μνήμης.
Τα αποθηκευτικά μέσα όπως οι σκληροί δίσκοι συνήθως έχουν μια μικρή cache (σε σύγχρονους εμπορικούς σκληρούς δίσκους φτάνει ακόμα και τα 32MB) που κρατά και αυτή δεδομένα. Μια μικρή περιγραφή αυτής της cache υπάρχει στην ενότητα 1.2.1. Η cache αυτή βοηθάει στην αύξηση των I/O επιδόσεων αλλά δεν θα αναφερθούμε περαιτέρω σε αυτήν καθώς είναι αρκετά μικρή και επιπλέον το λειτουργικό σύστημα δεν έχει τρόπο να την διαχειριστεί. Η διαχείρισή της ανήκει εξ ολοκλήρου στον σκληρό δίσκο και πιο συγκεκριμένα στον ελεγκτή του σκληρού δίσκου. Ο μόνος έλεγχος που διαθέτει λειτουργικό σύστημα είναι να αναγκάσει τον σκληρό δίσκο να γράψει τα δεδομένα που υπάρχουν στην cache του, στην μαγνητική του μονάδα αποθήκευσης.

Κεφάλαιο 3.
Συστήματα Αποθήκευσης στο Linux

3.1 Εισαγωγή

Στόχος του κεφαλαίου αυτού είναι να γίνει μια παρουσίαση των αρχών βάση των οποίων το λειτουργικό σύστημα Linux χειρίζεται τις συσκευές αποθήκευσης.

3.2 Γενικά για το Linux

Το (Linux ή πιο σωστά το) GNU/Linux είναι ένα λειτουργικό σύστημα που ανήκει στην οικογένεια λειτουργικών συστημάτων τύπου UNIX. Αρχικά αναπτύχθηκε από τον τότε μεταπτυχιακό φοιτητή Linus Torvalds το 1991 ως ένας κλώνος του UNIX που θα έτρεχε στον επεξεργαστή 80386 της Intel. Σήμερα το Linux αναπτύσσεται από μια κοινότητα εκατοντάδων προγραμματιστών των οποίων τις ενέργειες συντονίζει ο Torvalds. Μέσω της σκληρής δουλειάς αυτής της κοινότητας, το Linux έχει μεταφερθεί και στις παρακάτω αρχιτεκτονικές: x86_64, Alpha, Itanium, MIPS, PowerPC, SPARC κ.α.

Ένα σημαντικό πλεονέκτημα που έχει το Linux έναντι άλλων λειτουργικών συστημάτων είναι ότι εκδίδεται κάτω από την GPL άδεια, πράγμα που πρακτικά σημαίνει ότι ο πηγαίος κώδικας του είναι ανοιχτός και οποιοσδήποτε το επιθυμεί είναι ελεύθερος να τον μελετήσει και να τον τροποποιήσει.

Το Linux αποτελείται από τον πυρήνα και ένα σύνολο εφαρμογών (που εκδίδονται και αυτές υπό την άδεια GPL) παρέχοντας έτσι ένα πλήρως λειτουργικό και σύγχρονο UNIX σύστημα. Ο πυρήνας του Linux είναι μονολιθικός που σημαίνει ότι όλα τα τμήματά του τρέχουν στο επίπεδο πυρήνα σε ένα ενιαίο χώρο διευθύνσεων (σε αντίθεση με την αρχιτεκτονική microkernel όπου τα συστήματα αρχείων, οι οδηγοί συσκευών και οι υλοποιήσεις των δικτυακών πρωτοκόλλων τρέχουν ως εφαρμογές επίπεδο χρήστη). Η ύπαρξη μονολιθικού πυρήνα έχει το πλεονέκτημα ότι συνεπάγεται αυξημένες επιδόσεις σε σχέση με άλλες αρχιτεκτονικές όπως microkernel (αν και τα τελευταία χρόνια ορισμένοι microkernels όπως οι QNX και L4 συγκρίνονται σε ταχύτητα με μονολιθικούς πυρήνες). Το μειονέκτημα της ύπαρξης μονολιθικού πυρήνα είναι η αυξημένη συνεκτικότητα των τμημάτων του (η συνεκτικότητα εδώ εννοείται με την ορολογία της τεχνολογίας λογισμικού). Αυτό έχει ως αποτέλεσμα το προγραμματιστικό περιβάλλον στο kernel space είναι δύσκολο στην χρήση.
Όπως θα δούμε και λίγο παρακάτω, ο πυρήνας του Linux έχει την δυνατότητα να επεκταθεί με την χρήση των modules.

Τα πιο σημαντικά πλεονεκτήματα που έχει να προσφέρει το Linux είναι τα εξής:

- Είναι δωρεάν. Μπορεί οποιοσδήποτε εγκαθιστώντας το Linux σε έναν υποστηριζόμενο υπολογιστή να έχει ένα πλήρως λειτουργικό και δωρεάν UNIX σύστημα.

- Έχει πάρα πολλές δυνατότητες παραμετροποίησης. Με την χρήση των επιλογών που δίνονται κατά την διαδικασία μεταγλώττισης του πυρήνα, μπορούν να επιλεχθούν μόνο τα χαρακτηριστικά εκείνα που ενδιαφέρουν τον εκάστοτε χρήστη. Για παράδειγμα μπορούν να επιλεγούν ποιοι οδηγοί συσκευών θα υπάρχουν στον πυρήνα και ποιοι όχι. Επιπλέον, άτομα που διαθέτουν τις απαραίτητες γνώσεις μπορούν να τροποποιήσουν μόνοι τους τον πυρήνα αφού είναι διαθέσιμος ο πηγαίος κώδικάς του.
- Ο πυρήνας του έχει μικρό μέγεθος καθώς πολλές δυνατότητες έχουν μεταφερθεί στα modules που είναι κομμάτια κώδικα που μπορούν να φορτωθούν δυναμικά (δηλαδή κατά τον χρόνο εκτέλεσης) στον πυρήνα.
- Τρέχει ακόμα και σε παλαιό hardware. Ο πυρήνας του Linux μπορεί να τρέξει ακόμα και σε επεξεργαστές 386 και με RAM των 4MB. Για να τρέξουν σύγχρονες εφαρμογές απαιτείται και σύγχρονο hardware, αλλά αυτές είναι απαιτήσεις των εφαρμογών και όχι του ίδιου του λειτουργικού.
- Έχει πολύ καλές επιδόσεις. Τα συστήματα που τρέχουν Linux πετυχαίνουν υψηλές επιδόσεις χάρη στην πλήρη εκμετάλλευση του υποκείμενου hardware. Ο κύριος στόχος του Linux είναι να παρέχει υψηλές επιδόσεις και για αυτό το λόγο διάφορα υποσυστήματα τα οποία απαντώνται σε άλλα λειτουργικά συστήματα τύπου UNIX και επιβάλλουν μεγάλο overhead, όπως το STREAMS I/O υποσύστημα, απουσιάζουν εντελών από το Linux.
- Παρέχει απαράμιλλη συμβατότητα. Η συμβατότητα αυτή χωρίζεται σε τρία μέρη.

  Πρώτον το Linux δίνει την δυνατότητα στους χρήστες του να χρησιμοποιούν διάφορα συστήματα αρχείων που προορίζονται για διαφόρων ειδών συσκευές και απαντώνται σε διάφορα λειτουργικά συστήματα.
  Τα συστήματα αρχείων που υποστηρίζει το Linux είναι: Τα ext2, ext3 και ReiserFS που οφείλουν την ύπαρξή τους στο Linux.
  Τα sysv, UFS, MINIX και Veritas VxFs που είναι συστήματα αρχείων που αναπτύχθηκαν για άλλα τύπου UNIX λειτουργικά συστήματα.
  Τα MSDOS, VFAT και NTFS συστήματα αρχείων που χρησιμοποιούνται σε λειτουργικά συστήματα της Microsoft.
  ISO9660 και UDF που είναι τα συστήματα αρχείων που χρησιμοποιούνται σε δίσκους CD και DVD αντίστοιχα.
  Διάφορα συστήματα αρχείων που ανήκουν σε εταιρείες λογισμικού όπως τα JFS, XFS, HPFS, HFS, AAFS, ADFS.
  Δικτυακά συστήματα αρχείων όπως τα NFS, Coda, AFS, CIFS, NCP, GFS και OCFS2.
  Συστήματα αρχείων που δεν κάνουν ουσιαστικά διαχείριση αποθηκευτικού χώρου, αλλά εξυπηρετούν κάποιον ειδικό σκοπό. Αυτά είναι τα procfs, sysfs, debugfs κ.α.

  Εδώ πρέπει να αναφέρουμε πως τα συστήματα αυτά αρχείων υποστηρίζονται εγγενώς από το Linux, δηλαδή η υλοποίησή τους στον χώρο του πυρήνα (συνήθως με την μορφή module) παρέχεται στις επίσημες Linux πυρήνες που διανέμει το kernel.org. Βεβαίως στο Linux μπορούν να εγκατασταθούν κι άλλα συστήματα αρχείων που έχουν γραφτεί ειδικά για αυτό.

  Δεύτερον ο πυρήνας του Linux υλοποιεί πάρα πολλά δικτυακά πρωτόκολλα όπως: όλες τις εκδόσεις του Ethernet, FDDI, 802.11 (Wi-fi), 802.15

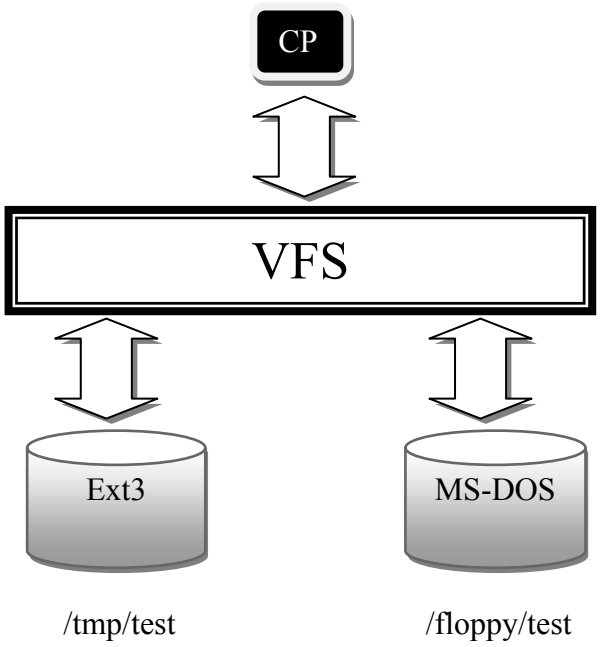
(Bluetooth) και Token Ring για να αναφέρουμε μόνο τα πρωτόκολλα του επιπέδου ζεύξης.

Τρίτον, χρησιμοποιώντας κατάλληλες βιβλιοθήκες, είναι δυνατή η εκτέλεση προγραμμάτων που είναι γραμμένα για άλλα λειτουργικά συστήματα, όπως MS-DOS, Windows, System V, 4.4BSD κ.α.

- Υποστηρίζεται πολύ καλά. Πάρα πολλές συσκευές διαθέτουν οδηγούς για το Linux και επιπλέον η έκδοση αυτών των οδηγών γίνεται γρήγορα και ανά τακτά χρονικά διαστήματα για να διορθωθούν τυχόν λάθη.

3.3 Linux VFS (Virtual File System)

Το VFS είναι ένα υποσύστημα του πυρήνα του Linux που έχει σχεδιαστεί ώστε να παρέχει ένα αφαιρετικό επίπεδο πάνω από τα πραγματικά συστήματα αρχείων.
Η χρήση του VFS δίνει την δυνατότητα στις εφαρμογές που τρέχουν στο Linux να έχουν πρόσβαση σε αρχεία που ανήκουν σε διαφορετικά συστήματα αρχείων με τον ίδιο ακριβώς τρόπο, δηλαδή χρησιμοποιώντας τις ίδιες κλήσεις συστήματος. Έτσι είναι δυνατή η χωρίς περαιτέρω ρυθμίσεις χρησιμοποίηση πολλών διαφορετικών συστημάτων αρχείων, ακόμα και εκείνων που είναι σχεδιασμένα για άλλα λειτουργικά συστήματα, όπως τα NTFS και VFAT που χρησιμοποιούνται τα λειτουργικά συστήματα Windows.
Για να δούμε στην πράξη το πλεονέκτημα που δίνει η ύπαρξη του VFS, εξετάζουμε το παρακάτω παράδειγμα.
Έστω ότι θέλουμε να αντιγράψουμε ένα αρχείο που βρίσκεται σε μια δισκέτα στην οποία έχει δημιουργηθεί το σύστημα αρχείων MS-DOS, σε ένα κατάλογο του συστήματός μας, που δουλεύει με σύστημα αρχείων ext3. Το πρόγραμμα που θα κάνει την αντιγραφή (δηλαδή το cp) δεν χρειάζεται να γνωρίζει ούτε το σύστημα αρχείων το οποίο υπάρχει στην δισκέτα, ούτε το σύστημα αρχείων που υπάρχει στον δίσκο. Μπορεί απλά γνωρίζοντας τα ονόματα των αρχείων και κάνοντας τα σωστά system calls να επιτύχει την αντιγραφή. Το γεγονός αυτό φαίνεται στην παρακάτω εικόνα.



Σχήμα 3.1: Παράδειγμα χρήσης VFS

Η παροχή υποστήριξης σε πολλά διαφορετικά συστήματα αρχείων είναι εφικτή επειδή το VFS παρέχει ένα συγκεκριμένο interface που πρέπει να υλοποιήσει κάθε σύστημα αρχείων που θα λειτουργήσει στο Linux. Το VFS δηλαδή από μόνο του δεν παρέχει υλοποίηση των system calls που αφορούν το σύστημα αρχείων, αλλά ο ρόλος του είναι να καλεί τις αντίστοιχες συναρτήσεις των κατάλληλων συστημάτων αρχείων όταν γίνεται προσπέλαση στα αρχεία τους.

Το VFS αν και δημιουργεί ένα μοντέλο που είναι ικανό να αναπαραστήσει κάθε σύστημα αρχείων (και ως εκ τούτου οποιοδήποτε σύστημα αρχείων μπορεί να υλοποιηθεί για να λειτουργεί με το VFS), εντούτοις ανακλά την σχεδιαστική φιλοσοφία των συστημάτων αρχείων του περιβάλλοντος UNIX. Αυτό σημαίνει ότι το VFS χρησιμοποιεί συγκεκριμένες δομές δεδομένων για να αναπαριστά τα αρχεία και το σύστημα αρχείων και η υλοποίηση κάθε συστήματος αρχείων που είναι προορισμένο να λειτουργεί με το VFS, θα πρέπει να μπορεί να δημιουργήσει ανά πάσα στιγμή αυτές τις δομές. Για παράδειγμα υπάρχουν πολλά συστήματα αρχείων που δεν λογίζουν τους καταλόγους ως αρχεία και χρησιμοποιούν έναν πίνακα FAT (File Allocation Table) που αποθηκεύει την θέση κάθε αρχείου μέσα στον κατάλογο του συστήματος αρχείων. Οι Linux υλοποιήσεις αυτών των συστημάτων αρχείων θα πρέπει να μπορούν να δημιουργούν ανά πάσα στιγμή τις δομές δεδομένων που αναπαριστούν τον κατάλογο ως αρχείο χρησιμοποιώντας τις άλλες δομές που διαθέτουν.

Οι δομές δεδομένων που χρησιμοποιεί το VFS για την αναπαράσταση των αρχείων και του συστήματος αρχείων είναι οι εξής:

Superblock

Η δομή αυτή αποθηκεύει δεδομένα που αφορούν ένα ενεργό (δηλαδή ένα mounted)

σύστημα αρχείων, όπως το μέγεθος block του συστήματος αρχείων, το μέγιστο επιτρεπόμενο μέγεθος των αρχείων, μια λίστα των αρχείων του συστήματος αρχείων κ.α. Η δομή αυτή βρίσκεται αποθηκευμένη στον δίσκο αλλά αντιγράφεται στην μνήμη όταν γίνεται mount το σύστημα αρχείων.

Inode

Υπάρχει ένα inode για κάθε αρχείο του συστήματος αρχείων και αποθηκεύει πληροφορίες σχετικές με το αρχείο αυτό. Μερικές από τις πληροφορίες που αποθηκεύει είναι, το μοναδικό αναγνωριστικό του αρχείου, το μέγεθος του αρχείο κ.α. Πρέπει να επισημανθεί ότι το όνομα του αρχείου **δεν** αποθηκεύεται σε αυτή τη δομή. Το inode αποθηκεύεται στον δίσκο αλλά μπαίνει σε μια ειδική cache (την inode cache) που διαχειρίζεται ο πυρήνας όταν διαβαστεί το αρχείο, προκειμένου να αυξάνεται η ταχύτητα πρόσβασης σε αυτή αν χρειαστεί να γίνει εκ νέου προσπέλαση στο αρχείο.

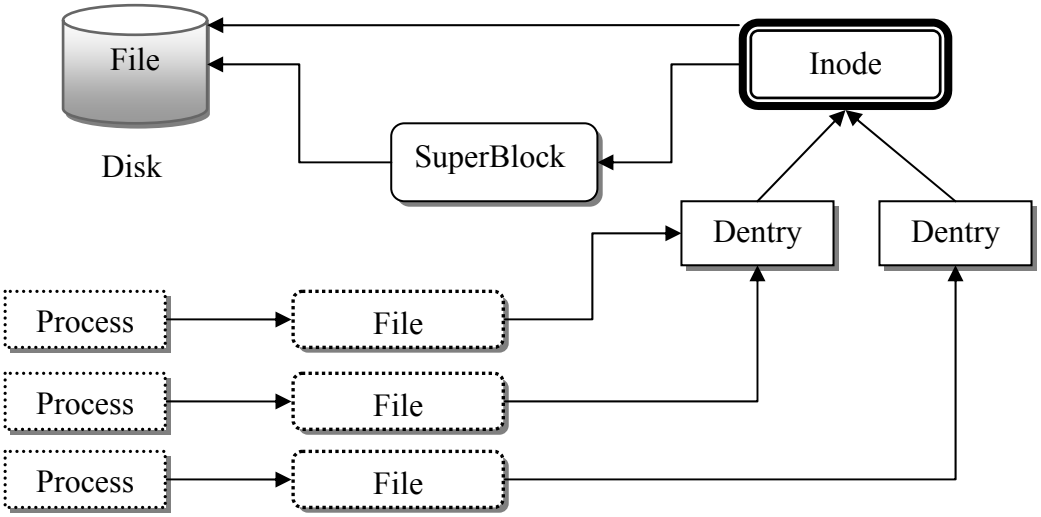
File

Αυτή η δομή κρατά δεδομένα σχετικά με την αλληλεπίδραση ενός ανοιχτού αρχείου και της διεργασίας που το άνοιξε. Για κάθε διεργασία που ανοίγει το αρχείο υπάρχει ξεχωριστή δομή file. Η δομή file υπάρχει μόνο στην μνήμη που διαχειρίζεται ο πυρήνας (και όχι στον δίσκο) και μόνο για όσο χρόνο η διεργασία έχει ανοιχτό το αρχείο.

Dentry

Η δομή dentry υπάρχει για να αντιστοιχεί το όνομα κάθε αρχείου με το μοναδικό αναγνωριστικό του. Οι πληροφορίες αυτές είναι αποθηκευμένες στον δίσκο, αλλά μπαίνουν σε μια ειδική cache (την dentry cache) που διαχειρίζεται ο πυρήνας όταν διαβαστεί το αρχείο, προκειμένου να αυξάνεται η ταχύτητα πρόσβασης σε αυτήν αν χρειαστεί να γίνει εκ νέου προσπέλαση στο αρχείο. Η δομή dentry ενός αρχείου περιέχει μεταξύ των άλλων και έναν δείκτη προς το inode του αρχείου.

Η παρακάτω εικόνα διασαφηνίζει την χρήση όλων των δομών που περιγράψαμε παραπάνω.



Σχήμα 3.2: Χρήση δομών δεδομένων του VFS

Στην εικόνα αυτή βλέπουμε πως τρεις διεργασίες χρησιμοποιούν ένα μόνο αρχείο. Το γεγονός αυτό το αντιλαμβανόμαστε καθώς βλέπουμε πως υπάρχει μόνο μια inode δομή δεδομένων που εμπλέκεται. Παρ' όλα αυτά εμπλέκονται πολλές άλλες δομές δεδομένων στον χειρισμό του αρχείου αυτού.
Κάθε διεργασία χρησιμοποιεί ένα file descriptor για να έχει πρόσβαση στο αρχείο. Το file descriptor αυτό, είναι ουσιαστικά ο δείκτης που αντιστοιχεί στη δομή file της διεργασίες, του πίνακα των ανοιχτών αρχείων (δηλαδή του πίνακα από δομές file) της διεργασίας. Κάθε διεργασία έχει τον δικό της πίνακα ανοιχτών διεργασιών και συνεπώς κάθε μία χρησιμοποιεί την δική της δομή file για να περιγράψει το αρχείο. Στη συνέχεια βλέπουμε πως η δομή file περιλαμβάνει έναν δείκτη προς την δομή dentry του αρχείου. Βλέπουμε ότι χρησιμοποιούνται δύο δομές dentry, πράγμα που σημαίνει πως το αρχείο προσπελάζεται από διαφορετικό link από την τρίτη διεργασία. Επειδή το link οδηγεί στο ίδιο αρχείο και οι δύο dentry δομές δείχνουν μέσω ενός δείκτη, την ίδια inode δομή. Με τη σειρά της η inode δομή του αρχείου, περιέχει ένα δείκτη προς το superblock του συστήματος αρχείων που το περιέχει καθώς και δείκτες για τις θέσεις των δεδομένων στο δίσκο.

3.4 Αρχεία Συσκευών στο Linux

Στα λειτουργικά συστήματα τύπου UNIX χρησιμοποιείται η έννοια του αρχείου προκειμένου να αναπαραστήσει πολλών ειδών πληροφορίες. Εξ ου και η γνωστή ρήση «Everything is a file!». Μια τέτοια αναπαράσταση είναι η χρήση του αρχείων για την αναπαράσταση των συσκευών του υπολογιστή. Τα αρχεία αυτά ονομάζονται αρχεία συσκευών. Από προγραμματιστική σκοπιά, η χρήση αυτών των αρχείων από εφαρμογές επιπέδου χρήστη είναι η ίδια με την χρήση απλών αρχείων δεδομένων. Με αυτό τον τρόπο μπορεί να γίνει απευθείας προσπέλαση των συσκευών από τις εφαρμογές, παρακάμπτοντας το σύστημα αρχείων που μπορεί να υπάρχουν σε αυτές.

Ανάλογα με τα χαρακτηριστικά της συσκευής που αναπαριστούν, τα αρχεία συσκευών χωρίζονται σε δύο κατηγορίες.

Αρχεία που αναπαριστούν συσκευές character (character devices). Κύριο χαρακτηριστικό των συσκευών character είναι ότι προσπέλαση στα δεδομένα είτε δεν μπορεί να γίνει τυχαία αλλά μόνο σειριακά (όπως μια σειριακή θύρα), είτε ο χρόνος που απαιτείται για την τυχαία προσπέλαση εξαρτάται από την θέση των δεδομένων μέσα στη συσκευή.
Αρχεία που αναπαριστούν συσκευές block. Οι συσκευές block διαθέτουν δύο χαρακτηριστικά. Πρώτον τα δεδομένα τους μπορούν πάντα να προσπελαστούν τυχαία και ο χρόνος προσπέλασης είναι (σχεδόν) ίδιος για όλα τα δεδομένα και δεύτερον οι συσκευές αυτές μπορούν να φιλοξενήσουν συστήματα αρχείων.
Οι πιο κοινές συσκευές block είναι οι σκληροί δίσκοι και τα οπτικά μέσα αποθήκευσης.
Ένας πίνακας που περιλαμβάνει μερικές συσκευές και την αναπαράστασή τους με αρχεία συσκευών φαίνεται παρακάτω:

Αρχείο	**Τύπος Συσκευής**	**Συσκευή**	**Major Number**	**Minor Number**
/dev/fd0	block	Δισκέτα	2	0
/dev/hda	block	1ος σκληρός δίσκος	3	0
/dev/hda2	block	2ο partition του 1ου δίσκου	3	2
/dev/hdb	block	2ος σκληρός δίσκος	3	64
/dev/hdb3	block	3ο partition του 2ου δίσκου	3	67
/dev/ttyp0	char	Terminal	3	0
/dev/console	char	Console	5	1
/dev/lp1	char	Εκτυπωτής με παρ/λη σύνδεση	6	1
/dev/ttyS0	char	1η σειριακή θύρα	4	64
/dev/rtc	char	Ρολόι	10	135
/dev/null	char	Null Device	1	3

Πίνακας 3.1: Παραδείγματα αρχείων συσκευών

Πρέπει να αναφερθεί ότι για κάθε συσκευή block, υπάρχει ένας ελάχιστος αριθμός δεδομένων που μεταφέρονται σε κάθε προσπέλαση και εξαρτάται από τα φυσικά χαρακτηριστικά του μέσου. Το πλήθος αυτό των δεδομένων ονομάζεται τομέας (sector).

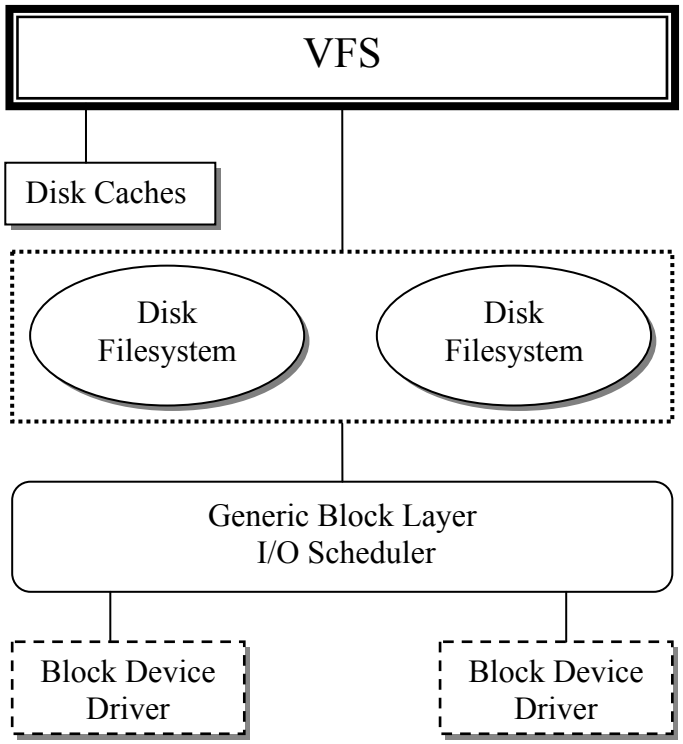
Το Linux επιτρέπει στις εφαρμογές να προσπελαύνουν οποιοδήποτε μέγεθος δεδομένων όταν χρησιμοποιούν τις συσκευές block, αλλά το ίδιο κάνει τις ανταλλαγές δεδομένων με την συσκευή σε μεγέθη δεδομένων ακέραια πολλαπλάσια του sector. Στο Linux η μικρότερη ποσότητα δεδομένων που μπορεί να μεταφερθεί από η προς μια συσκευή, ονομάζεται block δεδομένων και είναι ακέραιο πολλαπλάσιο του sector. Όταν μια εφαρμογή προσπελαύνει δεδομένα της συσκευής που έχουν μέγεθος μικρότερο το ενός block, το Linux κάνει την μεταφορά ολόκληρου του block, αλλά αντιγράφει στην περιοχή μνήμης της εφαρμογής μόνο όσα δεδομένα ζητήθηκαν.

Οι δικτυακές συσκευές δεν κατατάσσονται σε καμία από τις δύο κατηγορίες συσκευών που αναφέρθηκαν παραπάνω και για τον χειρισμό τους το Linux παρέχει ένα ξεχωριστό προγραμματιστικό interface.

Τα αρχεία συσκευών αναπαρίστανται στο Linux όπως και τα κανονικά αρχεία, δηλαδή με δομές dentry και inode, με την διαφορά ότι οι inode δομές τους στην θέση του δείκτη προς τα δεδομένα του αρχείο, έχουν έναν δείκτη προς ένα αναγνωριστικό της συσκευής. Το αναγνωριστικό αυτό αποτελείται από το τύπο της συσκευής καθώς και δύο αριθμούς τους major και minor number. Ο major number προσδιορίζει τον οδηγό της συσκευής ενώ ο minor number χρησιμοποιείται από τον οδηγό της συσκευής προκειμένου να μπορεί να ξεχωρίσει πολλαπλές συσκευές που πιθανόν διαχειρίζεται.

3.5 Χειρισμός των συσκευών Block στο Linux

Στο λειτουργικό σύστημα Linux, υπάρχει πολλά κομμάτια λογισμικού που εμπλέκονται στον χειρισμό των συσκευών block. Τα κομμάτια αυτά σχηματίζουν μια ιεραρχία που φαίνεται στην παρακάτω εικόνα.



Σχήμα 3.3: Λογισμικό του πυρήνα που εμπλέκεται στον χειρισμό block συσκευών

Ο χειρισμός των συσκευών block είναι καθοριστικός σε κάθε λειτουργικό σύστημα. Αυτό διότι οι συσκευές block αφενός έχουν πολύ υψηλούς χρόνους αναζήτησης των

δεδομένων, και αφετέρου παρέχουν υψηλούς ρυθμούς διαμεταγωγής όταν ολοκληρωθεί η αναζήτηση, πράγμα που κάνει τον χειρισμό τους εξαιρετικής σημασίας γεγονός για την επίτευξη υψηλών επιδόσεων.

Θα περιγράφουμε στη συνέχεια τις αρχές που χρησιμοποιεί το Linux για να χειριστεί τις συσκευές block.
Κάθε ενέργεια που αφορά συσκευές block απαιτεί την αλληλεπίδραση ενός μεγάλου αριθμού κομματιών του πυρήνα του Linux. Ας υποθέσουμε αρχικά ότι δημιουργείται μια αίτηση ανάγνωσης μιας συσκευής block (οι εγγραφές εξυπηρετούνται με παρόμοιο τρόπο). Οι ενέργειες που κάνει ο πυρήνας για να εξυπηρετήσει την αίτηση είναι οι εξής:

1. Αφού γίνει το context switching μετά την κλήση του read system call, ο έλεγχος περνάει στον πυρήνα. Αυτός καλεί αμέσως την συνάρτηση που χειρίζεται το read system call. Η συνάρτηση αυτή με την σειρά της καλεί την συνάρτηση ανάγνωσης του VFS. Το VFS είναι το ανώτερο στρώμα της αρχιτεκτονικής χειρισμού των συσκευών block και δημιουργεί ένα κοινό interface που πρέπει να υλοποιούν όλα τα συστήματα αρχείων που δουλεύουν στο Linux.

2. Στη συνέχεια το VFS ελέγχει αν τα ζητούμενα δεδομένα πρέπει να διαβαστούν από την block συσκευή ή αν υπάρχουν σε μια ειδική cache που διατηρεί ο πυρήνας, που ονομάζεται page cache. Στην περίπτωση που τα δεδομένα υπάρχουν στην cache, αρκεί να αντιγραφούν στην περιοχή μνήμης που δημιούργησε την αίτηση ανάγνωσης των δεδομένων, ολοκληρώνοντας τις ενέργειες που έχει να κάνει ο πυρήνας για να εξυπηρετήσει την αίτηση ανάγνωσης.

3. Για την συνέχεια υποθέτουμε πως τα απαραίτητα δεδομένα δεν βρίσκονται στην cache, συνεπώς ο πυρήνας πρέπει να τα διαβάσει από τον δίσκο. Για να γίνει αυτό, αρχικά θα πρέπει αρχικά να καθοριστεί η φυσική τους θέση στον δίσκο. Για να γίνει αυτό συνήθως απαιτούνται δύο βήματα:
   a) Καθορίζεται ο αριθμός των blocks του συστήματος αρχείων στα οποία αντιστοιχεί το μέγεθος των ζητούμενων δεδομένων. Όπως γνωρίζουμε το σύστημα αρχείων αποθηκεύει τα αρχεία σε σταθερού μεγέθους blocks δεδομένων.
   b) Στη συνέχεια καλείται μια συνάρτηση του συστήματος αρχείων στο οποίο ανήκει το αρχείο που θα προσπελαστεί, για να καθοριστούν τα φυσικά blocks του δίσκου στα οποία αντιστοιχούν τα δεδομένα του αρχείου που θα διαβαστούν. Όπως γνωρίζουμε, ο δίσκος λογίζεται ως μια σειρά φυσικών blocks όπου αποθηκεύονται δεδομένα. Επειδή μπορεί τα αρχεία να μην αποθηκεύεται σε συνεχόμενα blocks στον δίσκο, το σύστημα αρχείων παρέχει δομές που αντιστοιχούν τα δεδομένα κάθε αρχείου, σε blocks του δίσκου.

   Πρέπει να σημειώσουμε ότι αν η αίτηση ανάγνωσης δεδομένων δεν έγινε σε αρχείο που υπάγεται σε κάποιο σύστημα αρχείων, αλλά σε αρχείο συσκευής, τότε δεν υπάρχει καμία μετατροπή από blocks δεδομένων (άλλωστε δεν ορίζεται τέτοιο σε αυτή την περίπτωση) σε blocks δίσκου, αλλά απλά μετατρέπεται το offset της ανάγνωσης σε συγκεκριμένη θέση μέσα στην συσκευή.

4. Αφού πλέον έχουν καθοριστεί τα blocks του δίσκου τα οποία θα διαβαστούν, χρησιμοποιείται το generic block layer για να ξεκινήσει τις μεταφορές δεδομένων. Επειδή είναι πιθανό τα δεδομένα που ζητούνται από την ανάγνωση να μην βρίσκονται σε διπλανά block του δίσκου (να είναι δηλαδή το αρχείο κατακερματισμένο), το generic block layer μπορεί να ξεκινήσει πολλές διακριτές μεταφορές δεδομένων που θα εκτελεστούν ξεχωριστά.

5. Στη συνέχεια αναλαμβάνει δουλειά ο δρομολογητής I/O (I/O Scheduler). Η δουλειά του είναι να ταξινομεί τις τρέχουσες αιτήσεις για πρόσβαση στον δίσκο, ολοκληρώνοντας τες με μια σειρά που αντανακλά τις προεπιλεγμένες I/O πολιτικές του πυρήνα. Ουσιαστικά ο δρομολογητής I/O προσπαθεί να δρομολογεί τις αιτήσεις πρόσβασης στον δίσκο με μια σειρά που επιτυγχάνει ένα συγκεκριμένο ισοζύγιο ανάμεσα στις μεγαλύτερες δυνατές επιδόσεις και την δικαιοσύνη πρόσβασης στον δίσκο. Οι μεγαλύτερες επιδόσεις επιτυγχάνονται με το να συνενώνονται πολλές αιτήσεις για γειτονικά blocks του δίσκου, σε μία συνολική αίτηση η οποία θα υποβάλλεται στον δίσκο. Με την συνένωση πολλών γειτονικών αιτήσεων σε μία, μειώνεται στο ελάχιστο η επίδραση του χρόνου αναζήτησης των δεδομένων στον δίσκο αφού πλέον αρκεί να γίνει μόνο μία αναζήτηση για πολλές διαφορετικές αιτήσεις και εκμεταλλεύεται στο μέγιστο η δυνατότητα του δίσκου να παρέχει υψηλό ρυθμό διαμεταγωγής δεδομένων αφού μεταφέρονται πολλά δεδομένα μαζί. Για να αυξηθεί η πιθανότητα συνένωσης διαφορετικών αιτήσεων, ο δρομολογητής I/O δεν δρομολογεί απευθείας τις αιτήσεις, αλλά εισάγει μια τεχνητή καθυστέρηση με την ελπίδα ότι θα υπάρξουν εντός μικρού χρονικού διαστήματος νέες αιτήσεις προς γειτονικά δεδομένα. Η ελπίδα ότι θα υπάρξουν γρήγορα νέες προσβάσεις σε γειτονικά δεδομένα είναι βάσιμη καθώς πολλές προσβάσεις στα δεδομένα των δίσκων είναι σειριακές. Η δικαιοσύνη αφορά την δυνατότητα να έχουν όλες διεργασίες την δυνατότητα να προσπελαύνουν τον δίσκο σε λογικούς χρόνους. Αν δεν ληφθεί υπόψιν η παράμετρος της δικαιοσύνης στον δρομολογητή I/O, τότε υπάρχει πιθανότητα μια διεργασία να μην μπορεί να προσπελάσει τον δίσκο με ρυθμούς που να είναι έστω και ανεκτοί ενώ μια άλλη να εκμεταλλεύεται στο μέγιστο βαθμό τον ρυθμό διαμεταγωγής του δίσκου. Για παράδειγμα με τον δρομολογητή I/O της έκδοσης 2.4 του πυρήνα του Linux (ο λεγόμενος Linus Scheduler), όταν διαβάζεται συνεχόμενα ένα μεγάλο αρχείο από μια διεργασία, άλλες διεργασίες που διαβάζουν κάποιο άλλο αρχείο καθυστερούν πάρα πολύ. Στην έκδοση 2.6 του πυρήνα υπάρχουν τέσσερις διαφορετικοί δρομολογητές I/O, από τους οποίους μπορεί να διαλέξει ο χρήστης είτε κατά τον χρόνο φόρτωσης του πυρήνα (εισάγοντας την αντίστοιχη επιλογή στην καταχώριση του πυρήνα στο αρχείο /boot/grub/menu.lst), είτε κατά τον χρόνο εκτέλεσης (κάνοντας echo το όνομα του επιθυμητού δρομολογητή στο αρχείο /sys/block/*$devicename*/queue/scheduler όπου *$devicename* το όνομα της συσκευής έτσι όπως εμφανίζεται στον κατάλογο /dev).

   Πρέπει να αναφέρουμε επίσης ότι ο δρομολογητής I/O κρατάει μια συνδεδεμένη λίστα αιτήσεων για κάθε συσκευή αποθήκευσης που είναι συνδεδεμένη στον υπολογιστή και αναγνωρίζεται από το λειτουργικό σύστημα.

6. Το τελευταίο υποσύστημα του πυρήνα που εμπλέκεται στην διαδικασία που περιγράφουμε είναι ο οδηγός (driver) της συσκευής. Ο οδηγός λαμβάνει από τον δρομολογητή I/O την οδηγία να προσπελαστούν συγκεκριμένα blocks της συσκευής με αποτέλεσμα να στείλει στην συσκευή ειδικές εντολές για την πραγματοποίηση της μεταφοράς των δεδομένων. Στις περισσότερες περιπτώσεις η μεταφορά των δεδομένων γίνεται με την μέθοδο DMA (Direct Memory Access), στην οποία η ίδια η συσκευή αναλαμβάνει να μεταφέρει τα δεδομένα στην κεντρική μνήμη χωρίς παρεμβολή του επεξεργαστή. Οι θέσεις μνήμης στις οποίες θα αντιγραφούν τα δεδομένα, έχουν υποδειχθεί στην συσκευή από τον οδηγό της. Το DMA περιγράφεται αναλυτικά στην ενότητα 1.4.1.

3.6 Linux και Caching αρχείων

Το caching των αρχείων είναι πάρα πολύ σημαντικός παράγοντας στην επίτευξη υψηλών I/O επιδόσεων καθώς επαναλαμβανόμενες προσβάσεις στα ίδια δεδομένα είναι αρκετά συχνές. Είναι συχνές οι περιπτώσεις που μια διεργασία ζητά επανηλλημένα τα δεδομένα λίγων αρχείων, όπως και επίσης οι περιπτώσεις όπου πολλές διεργασίες ζητούν τα ίδια δεδομένα σε διαφορετικές χρονικές στιγμές.

Όταν γίνεται η προσπέλαση ενός αρχείου, είναι απαραίτητες οι δομές dentry και inode πέραν βεβαίως των δεδομένων του αρχείου. Όλες αυτές οι πληροφορίες βρίσκονται στο δίσκο, οπότε είναι προφανές ότι το caching των πληροφοριών αυξάνει σημαντικά τις επιδόσεις.
Το Linux κάνει εκτενή χρήση του caching των δεδομένων των αρχείων καθώς και των δομών δεδομένων που τα αναπαριστούν. Για τις inode δομές υπάρχει η inode cache, για τις dentry δομές υπάρχει η dentry cache και τέλος για τα δεδομένα των αρχείων υπάρχει η page cache.
Μια αλλαγή που υπάρχει από την έκδοση 2.4.10 του πυρήνα του Linux και μετά, είναι η κατάργηση του buffer cache για τα δεδομένα των αρχείων και η πλήρης αξιοποίηση της page cache για τον σκοπό αυτό.
Οι δύο πρώτες cache περιέχουν τις αντίστοιχες δομές δεδομένων για τα ίδια αρχεία. Δηλαδή, μια δομή inode για ένα αρχείο υπάρχει στην inode cache αν και μόνο αν υπάρχει στην dentry cache η αντίστοιχη dentry δομή για το ίδιο αρχείο. Το μέγεθος των cache αυτών δεν είναι συγκεκριμένο και εξαρτάται από το
μέγεθος της φυσικής μνήμης καθώς και από τις απαιτήσεις μνήμης που έχουν οι εφαρμογές που τρέχουν. Η πολιτική εξόδου από τις cache αυτές είναι Least
Recently Used (LRU). Για την γρήγορη εξακρίβωση της ύπαρξης ή όχι των δεδομένων στην cache και την ανάκτησή τους σε περίπτωση που όντως βρίσκονται, χρησιμοποιείται μια δομή hash. Η συνάρτηση hash δουλεύει πάνω στο όνομα του αρχείου για την dentry cache.
Η page cache είναι μια ειδική cache που αποθηκεύει δεδομένα αρχείων και δουλεύει με σελίδες δεδομένων όπως τις αντιλαμβάνεται το σύστημα εικονικής μνήμης
του πυρήνα. Παρόλο που η page cache δημιουργεί καταχωρίσεις σε μέγεθος σελίδων, εντούτοις επιτρέπει την αναζήτηση και προσπέλαση blocks δεδομένων ώστε να αξιοποιούνται οι ιδιότητες αποθήκευσης του συστήματος αρχείων.
Η cache αυτή δεν καταλαμβάνει συγκεκριμένο μέγεθος της μνήμης, αλλά αλλάζει δυναμικά ανάλογα με τις απαιτήσεις μνήμης των διεργασιών που τρέχουν.
Κάθε φορά που μια εφαρμογή θέλει να διαβάσει ή να γράψει δεδομένα από και προς

τον δίσκο αντίστοιχα, ελέγχεται πρώτα η page cache για να διαπιστωθεί αν υπάρχουν εκεί τα αντίστοιχα δεδομένα. Αν δεν υπάρχουν, τότε δημιουργείται μια νέα καταχώριση που θα περιέχει τα δεδομένα. Οι καταχωρίσεις αυτές καταλαμβάνουν χώρο που είναι ακέραιο πολλαπλάσιο των σελίδων δεδομένων (όπου σελίδα δεδομένων είναι μικρότερη ποσότητα δεδομένων με την οποία το σύστημα εικονικής μνήμης του πυρήνα διαχειρίζεται την μνήμη), δημιουργώντας έτσι εσωτερικό κατακερματισμό της page cache. Για την γρήγορη αναζήτηση δεδομένων στην page cache, δημιουργείται ένα radix tree που περιέχει δείκτες στις διάφορες καταχωρίσεις.
Η πολιτική εξόδου των καταχωρίσεων από την page cache είναι LRU.
Σε περιπτώσεις που ζητείται εγγραφή δεδομένων στον αποθηκευτικό χώρο, πρώτα γράφονται τα δεδομένα στην page cache και στη συνέχεια με μια μικρή καθυστέρηση στον δίσκο. Η λόγος που εισάγεται αυτή η καθυστέρηση έχει εξηγηθεί στην ενότητα 3.5.

Εδώ πρέπει να αναφέρουμε ένα άλλο σημαντικό πλεονέκτημα που επιφέρει η χρήση της page cache και βελτιώνει τις επιδόσεις του συστήματος. Τα δεδομένα που υπάρχουν σε αυτήν, όταν δεν έχουν υποστεί τροποποίηση δεν υπάρχει κανένας λόγος να γραφτούν στον δίσκο στην περίπτωση που χρειαστεί να εξέλθουν από την cache.
Για το λόγο αυτό δεδομένα που βρίσκονται στην page cache και δεν έχουν υποστεί τροποποίηση, είναι τα πρώτα που επιλέγονται από το σύστημα εικονική μνήμης για αποχώρηση από την μνήμη, όταν αυξηθούν από τις διεργασίες οι ανάγκες σε μνήμη. Η έξοδος αυτή των δεδομένων δεν προκαλεί καμία επιβάρυνση στο σύστημα, καθώς δεν γράφονται στον δίσκο. Αν αντίθετα είχαν επιλεγεί για αποχώρηση περιοχές μνήμης που ανήκαν σε διεργασίες, ενεργοποιούνταν δηλαδή το swapping, θα έπρεπε τα δεδομένα της διεργασίας που έβγαιναν από την μνήμη να γράφονταν στον δίσκο προκειμένου να μπορούν να ξαναμπούν στην μνήμη σε άλλη χρονική περίοδο.

Κεφάλαιο 4.
Συστήματα Αρχείων Κοινού Μέσου και άλλα παρόμοια Συστήματα Αρχείων

4.1 Εισαγωγή

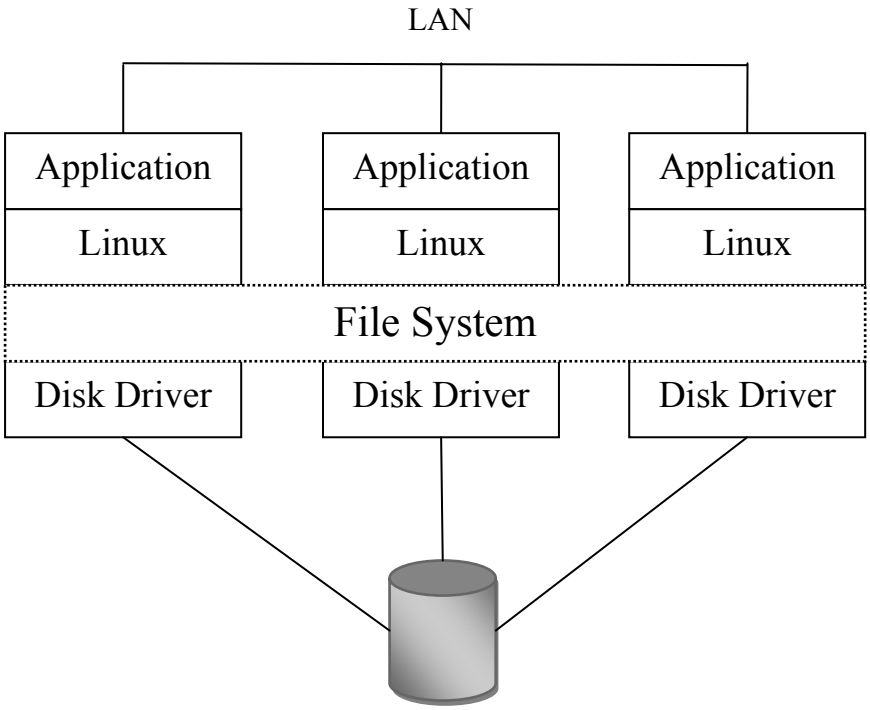
Τα συστήματα αρχείων που περιγράφουμε σε αυτό το κεφάλαιο εμπίπτουν στην γενικότερη κατηγορία των δικτυακών συστημάτων αρχείων. Στο κεφάλαιο αυτό θα περιγράψουμε τις διάφορες σχετικές κατηγορίες των συστημάτων αρχείων, καθώς και τα GFS και GPFS.

4.2 Συστήματα Αρχείων Κοινού Μέσου (Shared Disk File Systems)

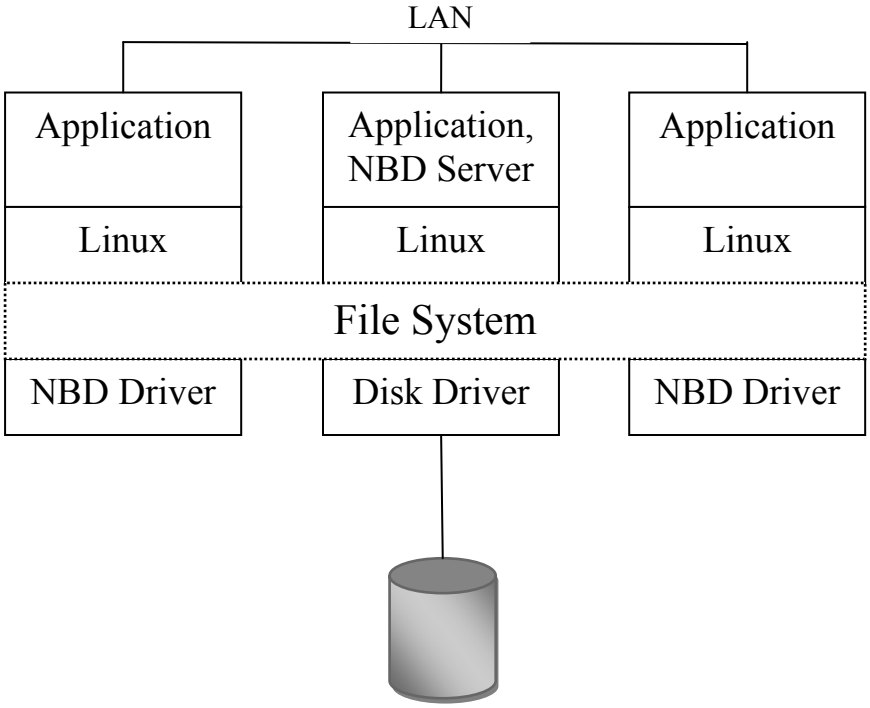
Τα Συστήματα Αρχείων Κοινού Μέσου (Shared Disk File System,Cluster File Systems) είναι συστήματα αρχείων που χρησιμοποιούνται σε περιβάλλοντα όπου πολλοί υπολογιστές έχουν κοινή πρόσβαση σε έναν ή περισσότερους δίσκους. Οι υπολογιστές που χρησιμοποιούν ένα σύστημα αρχείων κοινού μέσου σχηματίζουν συνήθως ένα cluster ενώ ο χώρος αποθήκευσης στον οποίο οι κόμβοι του cluster έχουν κοινή πρόσβαση αποτελείται από ένα SAN.

Στις περισσότερες περιπτώσεις οι κόμβοι ενός cluster έχουν όλοι άμεση πρόσβαση στον αποθηκευτικό χώρο, αφού συνδέονται στο SAN. Υπάρχουν όμως και περιπτώσεις που ορισμένοι κόμβοι του cluster δεν συνδέονται στο SAN. Για να έχουν και αυτοί πρόσβαση στον αποθηκευτικό χώρο, επικοινωνούν μέσω του δικτύου διασύνδεσης με άλλους κόμβους που έχουν άμεση πρόσβαση και αναλαμβάνουν τον ρόλο του File Server. Το μοντέλο πρόσβασης που χρησιμοποιούν οι κόμβοι χωρίς άμεση πρόσβαση στον αποθηκευτικό χώρο είναι το NBD μοντέλο του Linux.
Σε αυτό το μοντέλο ένας κόμβος (NBD Server) διαθέτει κάποιο σκληρό δίσκο και επιτρέπει σε άλλους κόμβους (NBD Clients) να τον χρησιμοποιούν χωρίς να έχουν άμεση πρόσβαση σε αυτόν. Αυτό είναι εφικτό καθώς ο NBD Server τρέχει μια εφαρμογή που δέχεται μέσω του δικτύου αιτήσεις πρόσβασης στον δίσκο από άλλους κόμβους και τις ικανοποιεί τοπικά. Οι NBD Client περιέχουν έναν οδηγό (driver) που επικοινωνεί με την εφαρμογή του NBD Server προκειμένου να της γνωστοποιήσει τις αιτήσεις πρόσβασης στον δίσκο που έγιναν από τον NBD Client. Ο driver ουσιαστικά δίνει στο λειτουργικό σύστημα την εντύπωση πως διαθέτει άλλον ένα τοπικά συνδεδεμένο δίσκο. Αυτό σημαίνει ότι το λειτουργικό χρησιμοποιεί το σύστημα αρχείων που διαθέτει ο πραγματικός δίσκος χωρίς να ξέρει ότι αυτός δεν είναι τοπικά συνδεδεμένος. Παρόλο που είναι εφικτή η ταυτόχρονη χρήση του συστήματος αρχείων του πραγματικού δίσκου και από τους NBD Client και από τον NBD Server, κάτι τέτοιο θα κατέστρεφε την ακεραιότητά του αν ήταν ένα τοπικό σύστημα αρχείων. Για ταυτόχρονη χρήση του δίσκου από τους κόμβους, θα πρέπει να έχει στηθεί πάνω του ένα σύστημα αρχείων κοινού μέσου και αυτό να χρησιμοποιείται από τα λειτουργικά συστήματα των κόμβων.

Παρακάτω φαίνονται στις εικόνες η περίπτωση άμεσης πρόσβασης στον αποθηκευτικό χώρο από όλους τους κόμβους και η περίπτωση χρήσης του NBD μοντέλου πρόσβασης αντίστοιχα.



Σχήμα 4.1: Απευθείας πρόσβαση στον αποθηκευτικό χώρο



Εικόνα 4.2: NBD πρόσβαση σε δίσκους ενός SAN

Οι NBD Clients αν και δεν έχουν άμεση πρόσβαση στον αποθηκευτικό χώρο, εντούτοις θεωρούνται ομότιμοι των NBD Servers όσον αφορά το σύστημα αρχείων.

Τα συστήματα αρχείων κοινού μέσου έχουν δύο κύριους σκοπούς.

- Να παρέχουν ανά πάσα στιγμή σε όλους τους κόμβους του cluster μια συνεπή εικόνα του συστήματος αρχείων (συνήθως ακολουθώντας την POSIX σημασιολογία, δηλαδή οι αλλαγές που κάνει ένας κόμβος είτε να φαίνονται στο σύνολό τους στους άλλους κόμβους) και
- Να παρέχουν σε όλους τους κόμβους ομοιόμορφη πρόσβαση σε ολόκληρο το εύρος του συστήματος αρχείων.

Ένας επιπλέον στόχος ενός συστήματος αρχείων κοινού μέσου είναι να επιτρέπει στις εφαρμογές που το χρησιμοποιούν, να προσπελαύνουν το σύστημα αρχείων μέσω του κοινού προγραμματιστικού interface. Για την περίπτωση του λειτουργικού συστήματος Linux, αυτό σημαίνει ότι τα συστήματα αρχείων κοινού μέσου θα πρέπει να λειτουργούν άψογα με VFS (Virtual File System) υλοποιώντας όλες τα system calls που αφορούν τα συστήματα αρχείων, όπως read, write, lseek, fstat κτλ.
Με αυτό τον τρόπο προγράμματα που έχουν γραφτεί χρησιμοποιώντας τις γνωστές αυτές κλήσεις συστήματος μπορούν να δουλέψουν χωρίς κανένα πρόβλημα πάνω από συστήματα αρχείων κοινού μέσου.

Πρέπει εδώ να σημειωθεί ότι σε αντίθεση με τα Παράλληλα Συστήματα Αρχείων που θα εξεταστούν παρακάτω, τα Συστήματα Αρχείων Κοινού Μέσου δεν στοχεύουν στο να παρέχουν τεράστιους ρυθμούς διαμεταγωγής στους κόμβους του cluster. Από πλευράς επιδόσεων, ο στόχος τους είναι να παρέχουν ρυθμούς διαμεταγωγής εφάμιλλους με αυτούς που πετυχαίνουν τα συστήματα αρχείων που λειτουργούν με τοπικά συνδεδεμένους δίσκους, όπως το ext3.

Σε γενικές γραμμές ένα σύστημα αρχείων κοινού μέσου δουλεύει ως εξής:
Όλοι οι κόμβοι που θα χρησιμοποιήσουν το σύστημα αρχείων θα πρέπει οπωσδήποτε να μπορούν να επικοινωνούν μεταξύ τους πάνω από το δίκτυο διασύνδεσης. Ανάλογα με το σύστημα αρχείων, είτε υπάρχουν ορισμένοι κόμβοι επιφορτισμένοι με τον ρόλο του διαχειριστή του συστήματος αρχείων είτε η διαχείριση αυτή γίνεται κατανεμημένα από όλους τους κόμβους. Η διαχείριση αυτή αφορά τα metadata του συστήματος αρχείων καθώς και τα metadata των αρχείων.

Για να διατηρείται η συνέπεια του συστήματος αρχείων, θα πρέπει οι κόμβοι να συντονίζουν τις προσβάσεις τους στα κοινά δεδομένα. Για τον συντονισμό αυτό χρησιμοποιείται κάποιος έλεγχος κλειδωμάτων. Ο έλεγχος αυτός είναι συνήθως κατανεμημένος για να αποφεύγονται οι μειωμένες επιδόσεις που θα προκαλούσε η ύπαρξη κεντρικού διαχειριστή κλειδωμάτων. Έτσι κάθε κόμβος που θέλει πρόσβαση σε ένα μέρος του συστήματος αρχείων θα πρέπει να λάβει το αντίστοιχο κλείδωμα. Ταυτόχρονες προσβάσεις άλλων κόμβων στο ίδιο μέρος του συστήματος αρχείων επιτρέπονται μόνο αν οι κόμβοι αυτοί μπορούν να λάβουν το ίδιο κλείδωμα. Για παράδειγμα πολλοί κόμβοι μπορούν να διαβάζουν ταυτόχρονα ένα αρχείο, αλλά δεν επιτρέπεται η πρόσβαση σε ένα αρχείο αν κάποιος κόμβος κατέχει κλείδωμα για εγγραφή στο αρχείο αυτό. Αν ένας κόμβος θέλει πρόσβαση σε ένα μέρος του συστήματος αρχείων και δεν μπορεί να λάβει το κλείδωμα θα πρέπει να περιμένει μέχρι ο κόμβος ή οι κόμβοι που κατέχουν τα κλειδώματα, να του το παραχωρήσουν.

Τέλος όταν ένας κόμβος προσπελαύνει ένα αρχείο, θα πρέπει να ενημερώνονται με κάποιο και τα metadata του αρχείου αυτού.

Τα συστήματα αυτά αρχείων συνήθως χρησιμοποιούν caching προκειμένου να αποφεύγονται οι προσβάσεις στον αποθηκευτικό χώρο όταν αυτό είναι εφικτό. Στα συστήματα αυτά όμως είναι απαραίτητο οι cache όλων των κόμβων να περιέχουν συνεπή δεδομένα. Αυτό επιτυγχάνεται και πάλι με τα κλειδώματα. Όταν ένας κόμβος λάβει ένα κλείδωμα για ένα μέρος του συστήματος αρχείων, τότε μπορεί να χρησιμοποιεί την cache για την προσπέλαση των αντίστοιχων δεδομένων. Όταν όμως παραδίδει το κλείδωμα, τα αντίστοιχα δεδομένα της cache πρέπει να ακυρωθούν και να γραφτούν στον αποθηκευτικό χώρο αν έχουν υποστεί τροποποίηση.

Για να αντιμετωπιστούν οι ασυνέπειες που προκύπτουν στο σύστημα αρχείων όταν κάποιος κόμβος τεθεί εκτός λειτουργίας χρησιμοποιούνται μηχανισμοί recovery βασισμένοι στα logs. Γενικά κάθε κόμβος πριν εκτελέσει ορισμένες ενέργειες που αφορούν το σύστημα αρχείων γράφει την περιγραφή της ενέργειας στο log. Το log βρίσκεται πάντα πλήρως ενημερωμένο στον αποθηκευτικό χώρο. Αφού ο κόμβος γράψει την περιγραφή της ενέργειας στο log, στη συνέχεια εκτελεί την ενέργεια.
Όταν ένας κόμβος τεθεί εκτός λειτουργίας τότε κάποιος άλλος κόμβος αναλαμβάνει να φέρει το σύστημα αρχείων σε συνεπή κατάσταση βασιζόμενο στο log του ανενεργού κόμβου.
Εδώ υπάρχει ένα λεπτό σημείο όσον αφορά τα logs. Έχει μεγάλη διαφορά αν θα μπαίνουν στα logs μόνο οι ενέργειες που αφορούν τα metadata, ή αν μπαίνουν και οι ενέργειες που αφορούν τα δεδομένα των αρχείων. Οι ενέργειες που αφορούν τα metadata σίγουρα μπαίνουν στα logs παρέχοντας εγγύηση για την συνέπεια του συστήματος αρχείων, αλλά αν δεν μπαίνουν στα logs και οι ενέργειες που αφορούν τα δεδομένα των αρχείων, τότε μπορεί μεν το σύστημα αρχείων να βρίσκεται σε συνεπή κατάσταση, αλλά τα δεδομένα κάποιων αρχείων να μην είναι.

Επίσης όταν ένας κόμβος τεθεί εκτός λειτουργίας θα πρέπει να υπάρχουν μηχανισμοί που εγγυώνται ότι ο ανενεργός κόμβος δεν έχει καμία πρόσβαση στον αποθηκευτικό χώρο, καθώς αν είχε θα μπορούσε όταν επανέλθει να προκαλέσει ζημιά στο σύστημα αρχείων.

Η παραπάνω περιγραφή συνεπάγεται πως υπάρχει μεγάλη ανάγκη επικοινωνίας μεταξύ των κόμβων. Η επικοινωνία αυτή επιτυγχάνεται πάνω από το δίκτυο διασύνδεσης χρησιμοποιώντας αυστηρά πρωτόκολλα επιπέδου εφαρμογής που εγγυώνται την συνεργασία των κόμβων.

Οι αποφάσεις που πρέπει να ληφθούν όσον αφορά την αρχιτεκτονική των συστημάτων αρχείων κοινού μέσου κατά την σχεδίαση και οι λεπτομέρειες της υλοποίηση των διαφόρων υποσυστημάτων τους, επηρεάζουν σε μεγάλο βαθμό την απόδοση, το scalling αλλά και την πολυπλοκότητα τους. Οι αποφάσεις αυτές αφορούν:

- την διαχείριση των μεταπληροφοριών(metadata) των αρχείων και του συστήματος αρχείων
- την διαχείριση των κλειδωμάτων
- την διαχείριση του caching
- την διαχείριση του recovery από απώλειες κόμβων μελών του cluster, δίσκων μελών του SAN αλλά και από προβλήματα που αφορούν το δίκτυο διασύνδεσης
- την δυνατότητα προσθήκης νέων κόμβων ή δίσκων σε χρονικές στιγμές εκτός της δημιουργίας του συστήματος αρχείων
- την δυνατότητα διαχείρισης και παραμετροποίησης του cluster και του

συστήματος αρχείων.

Υπάρχουν πολλά συστήματα αρχείων κοινού μέσου που διαφέρουν σε επιδόσεις, δυνατότητες scalling, πολυπλοκότητα και φυσικά τιμή!
Ορισμένα από αυτά είναι τα: GFS, OCFS2, CXFS.

4.3 Κατανεμημένα Συστήματα Αρχείων (Distributed File Systems)

Τα Κατανεμημένα Συστήματα Αρχείων είναι συστήματα αρχείων που επιλύουν διαφορετικές ανάγκες δικτυακής αποθήκευσης από ότι τα συστήματα αρχείων κοινού μέσου.
Στα κατανεμημένα συστήματα αρχείων οι κόμβοι που συμμετέχουν στο σύστημα αρχείων δεν αποτελούν ένα cluster και δεν διαθέτουν κάποιο κοινό αποθηκευτικό μέσο. Αντίθετα κάθε κόμβος συνεισφέρει μέρος του τοπικό του χώρου αποθήκευσης στο κατανεμημένο σύστημα αρχείων, δημιουργώντας με αυτόν τον τρόπο ένα εικονικό μέσο αποθήκευσης πάνω στο οποίο στήνεται το κατανεμημένο σύστημα αρχείων. Η δουλειά του κατανεμημένου συστήματος αρχείων είναι να επιτρέπει σε κάθε κόμβο να έχει πρόσβαση σε όλο τον εικονικό αποθηκευτικό χώρο, να διατηρεί την συνέπεια του συστήματος αρχείων και να καταβάλει την καλύτερη δυνατή προσπάθεια να διατηρεί ενεργό το σύστημα αρχείων όταν ορισμένοι κόμβοι παύουν να λειτουργούν ή η δικτυακή επικοινωνία με αυτούς καθίσταται αδύνατη.
Γνωστά Κατανεμημένα Συστήματα Αρχείων είναι τα AFS,DFS.

4.4 Παράλληλα Συστήματα Αρχείων (Parallel File Systems)

Τα Παράλληλα Συστήματα Αρχείων είναι και αυτά συστήματα αρχείων που εξυπηρετούν την πρόσβαση πολλών κόμβων σε κοινό αποθηκευτικό χώρο. Οι κύριες διαφορές των Παράλληλων Συστημάτων Αρχείων με τα Συστήματα Αρχείων Κοινού Μέσου είναι οι εξής:

- Δεν είναι συμμετρικά δηλαδή έχουν διακριτούς ρόλους Client και Server όσον αφορά την πρόσβαση στον αποθηκευτικό χώρο
- Χρησιμοποιείται εκτεταμένα η τεχνική του striping ώστε τα αρχεία να αποθηκεύονται κατά τμήματα σε πολλούς δίσκους (αν είναι διαθέσιμοι). Η τεχνική αυτή δίνει την δυνατότητα παράλληλης πρόσβασης σε πολλά (ενδεχομένως και όλα τα) τμήματα των αρχείων αυξάνοντας έτσι κατά πολύ τον ρυθμό διαμεταγωγής των δεδομένων των αρχείων προς τους κόμβους του cluster.
  Με αυτό τον τρόπο γίνεται αξιοποίηση των δίσκων του αποθηκευτικού χώρου όχι μόνο για την αύξηση του αποθηκευτικού χώρου αλλά και την αύξηση των επιδόσεων.
  Η τεχνική αυτή είναι παρόμοια σε σύλληψη με την τεχνική που χρησιμοποιείται στην διάταξη RAID 0 όπως περιγράψαμε στην αντίστοιχη ενότητα, με την διαφορά ότι η κατανομή των δεδομένων γίνεται από το ίδιο το σύστημα αρχείων και όχι από κάποιον ελεγκτή που κατανέμει συνεχόμενα λογικά blocks σε διαφορετικούς δίσκους.

Για να αξιοποιηθεί σωστά η τεχνική του striping θα πρέπει τα τμήματα των αρχείων

όχι μόνο ν' αποθηκεύονται σε διαφορετικούς δίσκους, αλλά θα πρέπει να αποθηκεύονται σε δίσκους που εξυπηρετούνται από διαφορετικούς servers αυξάνοντας με αυτό τον τρόπο την παραλληλία στην πρόσβαση των αρχείων.

Η τεχνική του striping και κατ' επέκταση τα παράλληλα συστήματα αρχείων είναι απαραίτητα προκειμένου να εκτελούνται πάρα πολύ γρήγορα οι I/O διαδικασίες σε cluster Υψηλής Υπολογιστικής Απόδοσης. Αυτά τα cluster μπορούν να έχουν τεράστια υπολογιστική ισχύ, αλλά αν οι I/O διαδικασίες δεν εκτελούνται γρήγορα, τότε οι συνολικές επιδόσεις τους είναι αισθητά μειωμένες.
Με την σωστή χρήση του striping μέσω ενός Παράλληλου Συστήματος Αρχείων, μπορεί να επιτευχθούν ρυθμοί διαμεταγωγής δεδομένων από και προς τον αποθηκευτικό χώρο, μεγαλύτερων από 100GB/sec (μέγεθος που έχουν επιτύχει τα παράλληλα συστήματα αρχείων GPFS, Luster και PVFS2). Ο ρυθμός αυτός είναι ικανοποιητικός για τις περισσότερες εφαρμογές που τρέχουν σε cluster Υψηλών Υπολογιστικών Επιδόσεων (τα cluster περιγράφονται στην ενότητα 5.5).
Τα πιο γνωστά τέτοια συστήματα αρχείων είναι το Lustre και το PVFS2.

4.5 Διαφορά Κατανεμημένων και Συστημάτων Αρχείων Κοινού Μέσου

Η ουσιώδης διαφορά μεταξύ των δύο τύπων συστημάτων αρχείων είναι ότι στα κατανεμημένα συστήματα αρχείων κάθε κόμβος έχει άμεση πρόσβαση μόνο στο μέρος του συστήματος αρχείων που βρίσκεται στον δικό του τοπικό χώρο αποθήκευσης ενώ η πρόσβαση στο υπόλοιπο σύστημα αρχείων γίνεται μέσω του δικτύου και εξυπηρετείται από τον εκάστοτε κόμβο που έχει τοπικό το μέρος του συστήματος αρχείων που ζητείται. Αντίθετα στα συστήματα αρχείων κοινού μέσου, όλοι οι κόμβοι του συστήματος έχουν άμεση και ομοιόμορφη πρόσβαση σε όλο το εύρος του συστήματος αρχείων.
Τα δύο είδη Συστημάτων Αρχείων χρησιμοποιούνται σε διαφορετικές περιπτώσεις. Τα κατανεμημένα συστήματα αρχείων χρησιμοποιούνται όταν διάφοροι κόμβοι που δεν μοιράζονται κανένα πόρο αλλά υπάρχει μεταξύ τους δικτυακή σύνδεση, θέλουν να προσπελαύνουν κοινά αρχεία. Τα συστήματα αρχείων κοινού μέσου χρησιμοποιούνται όταν πολλοί κόμβοι που έχουν πρόσβαση σε ένα κοινό μέσο αποθήκευσης όπως ένα SAN θέλουν κοινή πρόσβαση στον χώρο αυτό.

4.6 Διαφορά Στόχων Παράλληλων και Κοινού Μέσου Συστημάτων Αρχείων

Πρέπει να σημειωθεί ότι αν και τα παράλληλα και τα συστήματα αρχείων κοινού μέσου λειτουργούν και τα δύο πάνω σε δίκτυα SAN, εντούτοις έχουν διαφορετικούς στόχους.
Τα πρώτα χρησιμοποιούνται σε περιπτώσεις High Performace Computing (όπως είναι οι υπερυπολογιστές που συναντάει κανείς στην λίστα Top500) όπου είναι απαραίτητα τεράστια μεγέθη ρυθμών διαμεταγωγής από τους δίσκους προς τους κόμβους του cluster και αντίστροφα.
Τα δεύτερα χρησιμοποιούνται περισσότερο σε εταιρικά και ερευνητικά περιβάλλοντα όπου είναι απαραίτητα τα εξής: Πρώτον η κοινή πρόσβαση σε αρκετά μεγάλο όγκο δεδομένων με α) ταχύτητες παραπλήσιες εκείνων της πρόσβασης σε τοπικά συνδεδε-

μένα μέσα αποθήκευσης και β) δυνατότητες αποτροπής απωλειών δεδομένων. Δεύτερον η συνεχής λειτουργία ορισμένων υπηρεσιών (που είναι εφικτή με τη χρήση ειδικού λογισμικού που τρέχει πάνω στο σύστημα αρχείων). Σε αυτά τα περιβάλλοντα τα συστήματα αρχείων κοινού μέσου θα πρέπει να ικανοποιούν αποτελεσματικά τις δύο προηγούμενες απαιτήσεις και θα πρέπει να παρέχονται σε λογικό κόστος.

Ορισμένοι διακρίνουν και τα Συστήματα Αρχείων SAN ως ξεχωριστή κατηγορία. Η διαφορά τους σε σχέση με τα συστήματα αρχείων κοινού μέσου είναι ότι η διαχείριση των metadata γίνεται κεντρικά και συνεπώς τα συστήματα αυτά δεν μπορούν να χρησιμοποιηθούν στον ίδιο αριθμό κόμβων που μπορούν να χρησιμοποιηθούν τα συστήματα αρχείων κοινού μέσου.

4.7 Παραδείγματα Παράλληλων και Συστημάτων Αρχείων Κοινού Μέσου

4.7.1 GPFS

Το GPFS χαρακτηρίζεται ως ένα Παράλληλο Σύστημα Αρχείων Κοινού Μέσου (parallel, shared disk file system), δηλαδή αποτελεί ένα υβριδικό σύστημα αρχείων. Ακολουθεί την συμμετρική αρχιτεκτονική των συστημάτων αρχείων κοινού μέσου αλλά εκμεταλλεύεται στο έπακρο το πλήθος των δίσκων του κοινού αποθηκευτικού χώρου μοιράζοντας κομμάτια του αρχείου σε αυτούς, δηλαδή χρησιμοποιεί το striping. Το μέγεθος των κομματιών στα οποία χωρίζεται κάθε αρχείο εξαρτάται από το μέγεθος του αρχείου και δεν είναι το ίδιο με το block δεδομένων που χρησιμοποιεί το σύστημα αρχείων ως την μικρότερη μονάδα πρόσβασης στον δίσκο.

Παρ' όλο που το GPFS ακολουθεί συμμετρική αρχιτεκτονική δηλαδή δεν υπάρχουν Clients και Servers όσον αφορά την πρόσβαση στα δεδομένα του χώρου αποθήκευσης, εντούτοις υπάρχουν ορισμένοι ειδικοί ρόλοι που αναλαμβάνονται από συγκεκριμένους κόμβους. Υπάρχουν οι Primary και Secondary Cluster Managers, ορισμένοι File System Managers για κάθε σύστημα αρχείων του cluster και ένας ή περισσότεροι Token Managers για κάθε σύστημα αρχείων.
Η ύπαρξη συγκεκριμένων ρόλων σε αντίθεση με ότι που μπορεί να σκεφτεί κανείς αρχικά, δεν αποτελεί καθόλου ανασταλτικό παράγοντα στην επέκταση του GPFS για χρήση από πολλούς κόμβους. Αυτό διότι οι κόμβοι που αναλαμβάνουν τους ρόλους δεν κάνουν πολλή δουλειά οι ίδιοι, αλλά κατανέμουν τις απαραίτητες εργασίες στους κατάλληλους κόμβους του cluster.

Παρακάτω περιγράφονται συνοπτικά ορισμένες λεπτομέρειες για τον ρόλο του κάθε Server.

- Οι Cluster Managers διαχειρίζονται το cluster και ένας τουλάχιστον εκ των Primary και Secondary Cluster Manager είναι απαραίτητος για να είναι ενεργό το cluster.
- Για κάθε σύστημα αρχείων μέσα στο Cluster πρέπει να υπάρχει τουλάχιστον ένας File System Manager. Οι αρμοδιότητές του είναι να κάνει το configuration του συστήματος αρχείων όταν αυτό είναι απαραίτητο, να διαχειρίζεται την εκχώρηση ελεύθερου χώρου από το αποθηκευτικό χώρο

στους δίσκους κτλ. Αν κάποιος File System Manager τεθεί εκτός λειτουργίας τότε επιλέγεται κάποιος άλλος κόμβος στη θέση του ο οποίος αναλαμβάνει τα αντίστοιχα καθήκοντα.

- Τέλος ο Token Manager διαχειρίζεται την απόδοση και αποθήκευση της κατάστασης των κλειδωμάτων για όλο το σύστημα αρχείων. Για κάθε σύστημα αρχείων μπορούν να υπάρχουν περισσότεροι του ενός και επίσης μπορούν να αντικατασταθούν από κάποιον άλλο κόμβο αν τεθούν εκτός λειτουργίας.

Οι επιδόσεις του GPFS είναι πάρα πολύ καλές πράγμα που πιστοποιείται από μελέτες (βλ. Shared Parallel Filesystems in Heterogeneous Linux Multi-Cluster Enviroments και GPFS:A Shared-Disk File System for Large Computing Clusters) όσο και από το γεγονός ότι χρησιμοποιείται σε πολλούς από τους υπερυπολογιστές της λίστας Top500.

Τα δύο παρακάτω παραδείγματα υπερυπολογιστών στα οποία είναι εγκατεστημένο το GPFS πείθουν για το εξαιρετικό scalling που επιτυγχάνει. Και τα δύο έχουν κατασκευαστεί από την IBM και βρίσκονται στο Lawrence Livermore National Laboratory των Η.Π.Α.

- Ο υπερυπολογιστής Blue Gene/L οποίος είναι ο δεύτερος σε υπολογιστική ισχύ υπερυπολογιστής σύμφωνα με την λίστα του Ιουνίου 2008 του top500.org έχει 1664 I/O κόμβους, συνολικό μέγεθος αποθηκευμένων δεδομένων 1.89TB και μέγιστη επιτευγμένη διαμεταγωγή προς ένα ή περισσότερα αρχεία ταυτόχρονα 130GB/s.
- Ο υπερυπολογιστής ASC Purple ο οποίος βρίσκεται στην 21 θέση σύμφωνα με την ίδια λίστα, έχει πάνω από 10000 δίσκους, συνολικό μέγεθος αποθηκευμένων δεδομένων γύρω στα 2.8PB και η μέγιστη διαμεταγωγή που έχει επιτευχθεί προς ένα αρχείο ή πολλά αρχεία ταυτόχρονα είναι 92GB/s.

4.7.2 GFS

Το GFS αποτελεί ένα κλασσικό παράδειγμα ενός shared disk file system. Ήταν το πρώτο τέτοιο σύστημα αρχείων που εκδόθηκε κάτω από την άδεια GPL καθώς και το πρώτο που υποστήριξε αρχιτεκτονικές 64bit, με υποστήριξη για αρχιτεκτονικές x86, AMD64/EM64T και Itanium.

Στην τελευταία σταθερή έκδοσή του (6.1), χρησιμοποιώντας κατανεμημένο διαχειριστή κλειδωμάτων μπορεί να χρησιμοποιηθεί ταυτόχρονα από 300 κόμβους περίπου, σύμφωνα με τα επίσημα έγγραφα της κατασκευάστριας RedHat.

Αποτελεί μέρος του RedHat Cluster Suite και μαζί με τα άλλα μέρη αυτού του πακέτου προσφέρει υπηρεσίες όπως

- πρόσβαση σε κοινό αποθηκευτικό χώρο,
- υψηλή διαθεσιμότητα, δηλαδή την δυνατότητα να τρέχουν συνέχεια ορισμένες εφαρμογές ακόμα και αν ο κόμβος που τις φιλοξενεί τεθεί εκτός λειτουργίας, μεταφέροντας την λειτουργία της εφαρμογής σε άλλο κόμβο του cluster
- κατανομή φορτίου που έρχεται απ' έξω από το cluster, στους κόμβους του

cluster.

Ένας από τους στόχους του GFS είναι να παρέχει αυτές τις υπηρεσίες με τρόπο που είναι απλός στην διαχείριση.

Από αρχιτεκτονικής σκοπιάς στο GFS δεν υπάρχουν καθόλου ειδικοί ρόλοι και όλες οι εργασίες που πρέπει να επιτελεστούν μέσα στο cluster γίνονται κατανεμημένα. Το πιο άμεσα ανταγωνιστικό προς το GFS σύστημα αρχείων, είναι το OCFS2 της Oracle λόγω των κοινών στόχων και φιλοσοφιών που έχουν.

4.7.3 OCFS2

Το OCFS2 ανήκει στην ίδια ακριβώς κατηγορία προϊόντων με το GFS. Επιτρέπει σε πολλούς κόμβους που σχηματίζουν ένα cluster, παρέχοντας σε παράλληλες εφαρμογές που τρέχουν στους κόμβους του cluster την δυνατότητα να προσπελαύνουν διαφορετικά μέρη του συστήματος αρχείων ταυτόχρονα ενώ οι μη παράλληλες εφαρμογές μπορούν να χρησιμοποιήσουν το σύστημα αρχείων για να επιτύχουν υψηλή διαθεσιμότητα.

Τα κυριότερα χαρακτηριστικά του OCFS2 είναι:

- Δυνατότητα επιλογής ανάμεσα σε διάφορα μεγέθη block κατά την δημιουργία του συστήματος αρχείων.
- Ευέλικτη δέσμευση blocks δεδομένων(συνεχόμενη, μη συνεχόμενη, με κενά)
- Υποστήριξη logs για γρήγορο recovery
- Υποστήριξη πολλών αρχιτεκτονικών επεξεργαστών (x86, x86_64, ia64, ppc64)
- Κατανεμημένη διαχείριση κλειδωμάτων
- Υποστήριξη όλων των μορφών I/O που χρησιμοποιεί το Linux (Buffered, Direct, Asynchronous, Splice και Memory Mapped)
- Εύκολα στη χρήση διαχειριστικά εργαλεία

4.7.4 PVFS2

Το PVFS2 είναι ένα παράλληλο σύστημα αρχείων για Linux Clusters που είναι σχεδιασμένο να γίνεται scale αποδοτικά, να επιτρέπει την ύπαρξη πολύ μεγάλων συστήματα αρχείων της τάξης των PB και να παρέχει ρυθμούς διαμεταγωγής έως και 100GB/sec.

Για την λειτουργία του συστήματος αρχείων ορίζονται ορισμένοι κόμβοι ως servers του αποθηκευτικού χώρου. Οι κόμβοι αυτοί έχουν απευθείας πρόσβαση στον αποθηκευτικό χώρο και εξυπηρετούν τις αιτήσεις των υπολοίπων κόμβων (δηλαδή των clients) του cluster που δεν έχουν πρόσβαση στον αποθηκευτικό χώρο.

Για τον έλεγχο των metadata, το PVFS2 χρησιμοποιεί είτε κεντρικό server είτε κατανεμημένη διαχείριση από ορισμένους κόμβους. Η κύρια ευθύνη του/των metadata server(s) είναι γνωρίζει/ουν τον τρόπο με τον οποίο είναι κατανεμημένα τμήματα των αρχείων στους servers που εξυπηρετούν την I/O κίνηση.

Σε αντίθεση με τα άλλα συστήματα αρχείων που είδαμε προηγουμένως, το PVFS2 βασίζεται σε ένα τοπικό σύστημα αρχείων για την αποθήκευση των δεδομένων από τους servers.

Τα κύρια χαρακτηριστικά του PVFS2 είναι:

- Η επίτευξη υψηλών επιδόσεων σε περιβάλλοντα όπου υπάρχουν πολλές ταυτόχρονες προσπελάσεις σε μεγάλα τμήματα του αποθηκευτικού χώρου.
- Η modular σχεδίασή του επιτρέποντας την εύκολη παροχή υποστήριξης για νέο hardware και νέους αλγορίθμους.
- Η υποστήριξη πληθώρας hardware, καθώς μπορεί να λειτουργήσει σε διάφορες αρχιτεκτονικές όπως x86, IA64, Opteron, PowerPC, Alpha και MIPS.
- Η εγγενής υποστήριξη MPI-I/O.
- Η εύκολη εγκατάσταση.

Τέλος αξίζει να σημειωθούν δύο πράγματα όσον αφορά το PVFS2: Πρώτον ότι αποτελεί λογισμικό ανοιχτού κώδικα και εκδίδεται κάτω από την GPL άδεια και δεύτερον ότι δεν παρέχει καθόλου client-side caching με αποτέλεσμα κάθε αίτηση του client να πρέπει να εξυπηρετηθεί από κάποιον/κάποιους server.

4.8 Διαφορές GFS και GPFS

Στην ενότητα αυτή αναφέρουμε τις βασικές διαφορές των δύο συστημάτων αρχείων που μελετάμε. Οι διαφορές αυτές είναι διαφορές που προκύπτουν από την ανάγνωση documentation και papers που περιγράφουν αυτά τα συστήματα αρχείων.

Αν και ο βασικός ρόλος των δύο αυτών συστημάτων αρχείων είναι να παρέχουν σε όλους τους κόμβους του cluster άμεση και ομοιόμορφη πρόσβαση στον κοινό αποθηκευτικό χώρο, εντούτοις υπάρχουν πολλές και σημαντικές διαφορές στην σχεδίασή και στις λειτουργίες τους. Οι διαφορές αυτές παρατίθενται παρακάτω.

Χρήση Logic Volume Manager

Το GPFS δεν χρησιμοποιεί κάποιον Logic Volume Manager για να δημιουργεί λογικά partition αλλά διαχειρίζεται από μόνο του όλους τους δίσκους στήνοντας το σύστημα αρχείων κατευθείαν πάνω από αυτούς (αφού γίνουν γνωστοί στο cluster μέσω μιας ειδικής διαχειριστικής διαδικασίας). Αντίθετα το GFS είναι αναγκασμένο να χρησιμοποιεί το CLVM προκειμένου να μπορεί να δημιουργεί λογικά partition. Ένα GFS σύστημα αρχείων πρέπει να δημιουργηθεί πάνω σε ένα και μοναδικό λογικό partition που μπορεί να αντιπροσωπεύει πολλά φυσικά partition από διάφορους δίσκους.

Εφόσον το GPFS μπορεί να στηθεί πάνω σε πολλούς δίσκους μπορεί να αξιοποιήσει στο έπακρον την τεχνική του striping ώστε να αυξήσει κατά πολύ τον ρυθμό διαμεταγωγής δεδομένων των αρχείων από και προς τους κόμβους του cluster. Επειδή το GPFS κάνει striping από μόνο του, μπορεί να εκμεταλλευτεί τους δίσκους με τον καλύτερο δυνατό τρόπο ώστε να επιτύχει μεγαλύτερους ρυθμούς διαμεταγωγής,

ανοχή σε σφάλματα και καταμερισμό της I/O κίνησης στους δίσκους πάνω στους οποίους έχει δημιουργηθεί.
Το CLVM το οποίο χρησιμοποιεί το GFS έχει και αυτό την δυνατότητα να κάνει striping αλλά επειδή δουλεύει με λογικά blocks δεδομένων και όχι αρχεία, δεν μπορεί να κάνει βελτιστοποιημένο καταμερισμό της κίνησης προς τους δίσκους ώστε να αξιοποιούνται πλήρως για τις ανάγκες του συστήματος αρχείων.

Πρέπει να σημειώσουμε εδώ ότι ο τρόπος με τον οποίο χρησιμοποιούν τους πολλαπλούς δίσκους τα δύο συστήματα αρχείων, έχει αποτέλεσμα στο αν μπορούμε στην πράξη με tracing να δούμε την χρήση του striping. Στο GPFS όπως είπαμε δεν χρησιμοποιείται κάποιος Logic Volume Manager, συνεπώς όλοι οι δίσκοι φαίνονται στο λειτουργικό σύστημα στο οποίο υπάρχει το GPFS ως μια ξεχωριστή καταχώριση στον κατάλογο /dev του συστήματος. Αυτό έχει ως αποτέλεσμα να υπάρχει η δυνατότητα να είναι δυνατή η ταυτόχρονη καταγραφή των προσβάσεων στους δίσκους με τη χρήση του προγράμματος blktrace (που θα περιγραφεί στην ενότητα 7.4), ώστε να δούμε στην πράξη πως χρησιμοποιείται η τεχνική του striping από το GPFS.
Αντίθετα επειδή το GFS χρησιμοποιεί Logic Volume Manager, το λειτουργικό σύστημα που χρησιμοποιεί το GFS αντιλαμβάνεται έναν και μοναδικό δίσκο. Ενδεχόμενη καταγραφή πάνω στον δίσκο αυτό απλά θα φανέρωνε την προσπέλαση στον συνολικό λογικό χώρο, αλλά δεν θα φανέρωνε καμία λεπτομέρεια για την προσπέλαση σε συγκεκριμένους δίσκους.

Χρήση Αυτοματοποιημένων Μηχανισμών Διαχείρισης από το GPFS

Το GPFS διαθέτει μηχανισμούς που βοηθούν στην αυτοματοποιημένη διαχείριση των δεδομένων του cluster που λείπουν παντελώς από το GFS.
Οι μηχανισμοί αυτοί είναι πρώτον οι ομάδες δίσκων (storage pools), δεύτερον τα σύνολα αρχείων (filesets) που είναι υποδένδρα του δένδρου των καταλόγων του συστήματος αρχείων και τρίτον οι πολιτικές που καθορίζονται από τον χρήστη (user defined policies) ή τους χρήστες του συστήματος αρχείων.
Ο μηχανισμός των ομάδων δίσκων επιτρέπει στον διαχειριστή του συστήματος αρχείων να ομαδοποιεί δίσκους ανάλογα με τις επιδόσεις τους, την αξιοπιστία τους και την χωρική τοπικότητά τους. Για παράδειγμα θα μπορούσε να δημιουργηθεί μία ομάδα δίσκων που περιέχει πολύ γρήγορους δίσκους και μια άλλη ομάδα που περιέχει πιο αργούς δίσκους.
Χρησιμοποιώντας τον μηχανισμό των filesets ο διαχειριστής μπορεί να εφαρμόσει διαφορετικά user defined policies σε διαφορετικά σημεία του συστήματος αρχείων. Τα user defined policies έχουν να κάνουν με δύο πράγματα. Πρώτον, την τοποθέτηση και την διαχείριση των αρχείων. Η τοποθέτηση των αρχείων αναφέρεται σε ποια ομάδα δίσκων αποθηκεύονται αρχικά τα αρχεία.
Για παράδειγμα, θα μπορούσαν πολύ σημαντικά αρχεία να τοποθετηθούν σε μια ομάδα δίσκων έχει δημιουργηθεί με διάταξη RAID 1 για αυτόματη αντιγραφή των δεδομένων. Δεύτερον την διαχείριση των αρχείων. Η διαχείριση των αρχείων αναφέρεται στο πως γίνεται η μεταφορά, η αντιγραφή και η διαγραφή των αρχείων. Χρησιμοποιώντας μια γλώσσα που μοιάζει με την SQL είναι δυνατή η γρήγορη και αυτοματοποιημένη διαχείριση των αρχείων. Για παράδειγμα, θα μπορούσε να χρησιμοποιηθεί μια πολιτική που να λέει ότι όλα τα αρχεία που έχουν κατάληξη .odf κάτω από τον κατάλογο /perfomance_reports να αντιγράφονται αυτόματα στην ομάδα

δίσκων no.3.

Modular Σχεδίαση του GFS

Ένας από τους σχεδιαστικούς στόχους του GFS είναι να αποτελεί ένα modular κομμάτι κώδικα. Αυτό του επιτρέπει να προσαρμόζεται γρήγορα σε νέα υπολογιστικά περιβάλλοντα και να είναι εύκολα επεκτάσιμο. Αντίθετα το GPFS είναι ένα μονολιθικό λογισμικό που επιτελεί μόνο του όλες τις απαραίτητες λειτουργίες και όπως αποδεικνύεται τις επιτελεί πολύ αποτελεσματικά.

Λόγο της modular σχεδίασής του, το GFS δεν είναι πλήρως λειτουργικό αλλά βασίζεται σε άλλα υποσυστήματα που βρίσκονται στο πακέτο RedHat Cluster Suite προκειμένου να επιτελέσει ορισμένες λειτουργίες του.
Οι λειτουργίες τις οποίες δεν κάνει το GFS από μόνο του αλλά ανατίθενται σε άλλα υποσυστήματα του RedHat Cluster Suite, τα οποία βεβαίως μπορούν να αντικατασταθούν στο μέλλον με καλύτερα (αν υπάρξουν τέτοια) είναι οι εξής:

Διαχείριση κλειδωμάτων
Η διαχείριση κλειδωμάτων στην τελευταία έκδοση του GFS ανατίθεται στο σύστημα LOCK_DLM. Το LOCK_DLM αποτελεί έναν κατανεμημένο διαχειριστή κλειδωμάτων που βασίζεται όπως και πολλοί άλλοι κατανεμημένοι διαχειριστές κλειδωμάτων σε UNIX συστήματα στην σχεδίαση του πρώτου κατανεμημένου διαχειριστή κλειδωμάτων που εμφανίστηκε σε αυτά τα συστήματα, του VaxCluster DLM.
Το LOCK_DLM είναι ένα αυτόνομο λογισμικό που παρέχει ένα συγκεκριμένο interface, ώστε λογισμικό σε επίπεδο πυρήνα (όπως το GFS) αλλά και σε επίπεδο χρήστη να μπορεί να το χρησιμοποιεί για να κάνει κατανεμημένη διαχείριση κλειδωμάτων.

Διαχείριση του χώρου ονομάτων
Το GFS χρησιμοποιεί ένα αυτόνομο διαχειριστή του χώρου ονομάτων των μελών του cluster ώστε να μπορεί να ξεχωρίσει αυτά τα μέλη.

Recovery
Το recovery ενός GFS συστήματος αρχείων που είναι απαραίτητη όταν ένας κόμβος τεθεί εκτός λειτουργίας, γίνεται από το λογισμικό διαχείρισης του cluster. Το λογισμικό αυτό πρέπει να γνωρίζει ανά πάσα στιγμή όλους τους κόμβους του cluster που έχουν κάνει mount κάθε GFS σύστημα αρχείων ώστε να είναι δυνατή η επιλογή κάποιου ενεργού κόμβου για να κάνει recovery χρησιμοποιώντας το log (που βρίσκεται στον κοινό χώρο αποθήκευσης) του ανενεργού κόμβου.

Αποκλεισμός κόμβων από το κοινό μέσο αποθήκευσης
Η λειτουργία αυτή ονομάζεται fencing και για να μπορεί το GFS να συνεχίσει να λειτουργεί όταν κάποιος κόμβος τεθεί εκτός λειτουργίας, θα πρέπει το λογισμικό που διαχειρίζεται το cluster σε συνεργασία με το λογισμικό που κάνει με το fencing, να εγγυώνται στο GFS ότι ο ανενεργός κόμβος δεν έχει από εκείνη τη στιγμή και πέρα καμία απολύτως πρόσβαση στον κοινό χώρο αποθήκευσης.

Πρώτο mount
Όταν κάποιος κόμβος είναι ο πρώτος στο cluster που κάνει mount ένα GFS σύστημα αρχείων, θα πρέπει το λογισμικό διαχείρισης και το λογισμικό που αναλαμβάνει το fencing να μην επιτρέπουν σε κανένα άλλο κόμβο να κάνει mount το συγκεκριμένο σύστημα αρχείων πρωτού ο αρχικός κόμβος ολοκληρώσει όλες τις απαραίτητες ενέργειες αρχικοποίησης.

Διαχείριση Κλειδωμάτων

Τα δύο συστήματα αρχείων διαφοροποιούνται και στον τρόπο με τον οποίο διαχειρίζονται τα κλειδώματα που είναι απαραίτητα ώστε να διατηρείται η ακεραιότητα τους. Προκειμένου να επιτύχουν καλύτερες δυνατότητες επέκτασης και τα δύο κάνουν χρήση κατανεμημένου ελέγχου κλειδωμάτων (το GFS αν και έχει modular σχεδίαση και μπορεί να χρησιμοποιήσει διάφορους διαχειριστές κλειδωμάτων, στην τελευταία έκδοσή του συνίσταται η χρήση του dlm lock manager) αλλά παρουσιάζουν διαφορές στις λεπτομέρειες της διαχείρισης των κλειδωμάτων αυτών.

Πιο συγκεκριμένα τα κλειδώματα στο GPFS δουλεύουν ως εξής: Για κάθε σύστημα αρχείων υπάρχουν ένας ή περισσότεροι server που ονομάζονται token managers οι οποίοι κρατούν την πληροφορίες για την κατάσταση όλων των κλειδωμάτων και έχουν το ρόλο της απόδοσης τους προς τους κόμβους που θέλουν να χρησιμοποιήσουν το σύστημα αρχείων. Για να έχει πρόσβαση ένας κόμβος Α σε κάποιο μέρος του συστήματος αρχείων θα πρέπει να ζητήσει από τον token manager το αντίστοιχο κλείδωμα. Αν ο token manager δεν έχει αποδώσει σε κανέναν άλλο κόμβο το συγκεκριμένο κλείδωμα, τότε το παραχωρεί στον κόμβο Α. Από την στιγμή εκείνη και πέρα και μέχρι ο κόμβος Α να αφήσει το κλείδωμα και να το επιστρέψει στον token manager ή να το παραχωρήσει σε άλλο κόμβο Β ώστε να αποκτήσει αυτός την πρόσβαση στο σύστημα αρχείων που είχε ο Α, ο κόμβος Α είναι υπεύθυνος για το συγκεκριμένο κλείδωμα.
Αν την στιγμή που ο κόμβος Α ζητούσε το κλείδωμα από τον token manager το κλείδωμα αυτό το είχαν ένας ή περισσότεροι άλλοι κόμβοι τότε ο token manager θα επέστρεφε στον κόμβο Α την λίστα με τους κόμβους που κρατούν το κλείδωμα και θα έπρεπε ο κόμβος Α από μόνος του να τους ζητήσει την παραχώρηση του κλειδώματος. Βλέπουμε πως αν και υπάρχει κεντρικός token manager (που ενδέχεται να μετατραπεί σε κατανεμημένο αν ο αριθμός των κόμβοι που χρησιμοποιούν το σύστημα αρχείων ξεπεράσει ένα κατώφλι), η επικοινωνία των κόμβων μαζί του περιορίζεται σε δύο μηνύματα το πολύ για την αίτηση κλειδωμάτων και μηδέν ή ένα κατά την αποδέσμευσή. Ανυπαρξία μηνυμάτων προς τον token manager έχουμε στην περίπτωση που το κλείδωμα παραχωρηθεί σε άλλο κόμβο που όμως με τη σειρά του θα πρέπει να ενημερώσει τον token server ότι είναι ο νέος κάτοχος του κλειδώματος που κατείχε ο Α. Ένα μήνυμα είναι απαραίτητο στην περίπτωση που ο κόμβος απόδεσμεύσει το κλείδωμα και το επιστρέψει στον token manager.

Η διαχείριση των κλειδωμάτων στο GFS δουλεύει ως εξής: Όταν ένας κόμβος είναι ο πρώτος που προσπελαύνει κάποιο αρχείο, τότε αυτός γίνεται ο διαχειριστής του κλειδώματος στο συγκεκριμένο αρχείο και η κατάσταση αυτή γίνεται άμεσα γνωστή στους άλλους κόμβους που χρησιμοποιούν το ίδιο σύστημα αρχείων. Κάθε άλλος κόμβος που θέλει να έχει πρόσβαση στο συγκεκριμένο αρχείο θα πρέπει να

επικοινωνήσει με τον κόμβο που είναι διαχειριστής του κλειδώματος για το αρχείο για να ζητήσει το κλείδωμα. Συνεπώς στο GFS δεν υπάρχει μοναδικός διαχειριστής κλειδωμάτων. Κάθε κόμβος γίνεται διαχειριστής των κλειδωμάτων των αρχείων τα οποία προσπέλασε πρώτος.

Μια δεύτερη διαφορά στην διαχείριση των κλειδωμάτων έχει να κάνει με το πλήθος των δεδομένων που κλειδώνονται. Στο GFS κλειδώματα υπάρχουν κλειδώματα για ολόκληρα αρχεία μόνο, σε αντίθεση με το GPFS που μπορεί να κλειδωθεί από ένα block του αρχείου μέχρι ολόκληρο το αρχείο. Αυτό σημαίνει ότι στο GPFS κόμβοι που προσπελαύνουν διαφορετικά μέρη του ίδιου αρχείου δεν συγκρούονται για την απόκτηση κλειδωμάτων σε αντίθεση με το GFS.

Χειρισμός Caching

Αν και τα δύο συστήματα αρχείων εφαρμόζουν caching όταν προσπελάζονται αρχεία προκειμένου να αυξήσουν τις επιδόσεις τους, εντούτοις διαφέρον στον τρόπο που χειρίζονται την έξοδο των δεδομένων από την cache. Η διαφορά αυτή είναι άμεση συνέπεια του τρόπου που λειτουργούν οι διαχειριστές κλειδωμάτων τους.

Στο GFS επειδή μπορούν να κλειδωθούν μόνο ολόκληρα αρχεία, πρέπει κάθε φορά που ανακαλείται το κλείδωμα που κατέχει ένας κόμβος για ένα αρχείο, να ακυρώνονται όλα τα δεδομένα της cache που αφορούν το αρχείο ακόμα και αν ο επόμενος κόμβος που πήρε το κλείδωμα δεν αλλάξει παρά μόνο ένα byte του αρχείου. Αντίθετα επειδή το GPFS επιτρέπει την ύπαρξη κλειδωμάτων σε τμήματα του αρχείου, όταν ανακαλείται ένα κλείδωμα που αναφέρεται σε κομμάτι ενός αρχείου μόνο τα δεδομένα της cache που αντιστοιχούν σε αυτό το κομμάτι του αρχείου ακυρώνονται αφήνοντας τα υπόλοιπα δεδομένα του αρχείου που ενδεχομένως βρίσκονται στην cache ανέπαφα.

Σε όλες τις περιπτώσεις που είναι αναγκαία η ανάκληση κλειδώματα με συνέπεια την ακύρωση των αντίστοιχων δεδομένων της cache, τα δεδομένα αυτά πρέπει να γράφονται στον αποθηκευτικό χώρο, εφόσον έχουν υποστεί τροποποίηση εντός της cache (χρησιμοποιείται δηλαδή write behind).

Πρακτικά αποτελέσματα μέσω καταγραφών για την διαφορά αυτή στο caching θα παρουσιαστούν στο κεφάλαιο 8. Εκεί κάνοντας καταγραφή των προσβάσεων στον αποθηκευτικό χώρο μέσω του Blktrace, θα δούμε ότι αν ένας κόμβος έχει στην cache του τα δεδομένα ενός ολόκληρου αρχείου και κάποιος άλλος κόμβος αλλάξει μόνο ένα λίγα byte του αρχείου, τότε σε επόμενη ανάγνωση του πρώτου κόμβου στο αρχείο, αν το σύστημα αρχείων είναι το GPFS, θα αναγνωστεί μόνο το block δεδομένων στο οποίο ανήκαν τα αλλαγμένα byte. Τα υπόλοιπα δεδομένα διαβάζονται από την cache. Αν το σύστημα αρχείων είναι το GFS θα δούμε ότι πρακτικά διαβάζεται όλο το αρχείο από την αρχή.

Χρήση Fencing

Το GFS μπορεί να χρησιμοποιήσει πολλές μορφές fencing και είναι στην ευχέρεια του διαχειριστή του cluster να επιλέξει για κάθε κόμβο τον τρόπο με τον οποίο θα

αποκλειστεί η πρόσβασή του στο κοινό μέσο αποθήκευσης σε περίπτωση που καταστεί ανενεργός. Μερικές από τις μεθόδους που χρησιμοποιεί το GFS και μπορούν να εγγυηθούν τον αποκλεισμό ανενεργών κόμβων από το κοινό μέσο αποθήκευσης είναι 1) να δίνεται εντολή στην πρίζα απ' όπου τροφοδείται ο ανενεργός κόμβος με ρεύμα, να διακόψει την παροχή (ονομάζεται power fencing) και 2) να δίνεται εντολή στον fibre channel μεταγωγέα μέσω του οποίου έχει πρόσβαση στον κοινό χώρο αποθήκευσης ο ανενεργός κόμβος, να διακόψει στην σύνδεση προς τον κόμβο αυτό. Υπάρχουν αρκετές ακόμα μέθοδοι αλλά οι δύο παραπάνω είναι οι σημαντικότερες.

Το GPFS αφενός έχει μια και μοναδική μέθοδο μέσω της οποίας πραγματοποιείται fencing και αφετέρου η μέθοδος αυτή ακολουθεί τελείως διαφορετική προσέγγιση για τον αποκλεισμό των ανενεργών κόμβων. Η μέθοδος ονομάζεται disk leasing και λειτουργεί ως εξής: Κάθε κόμβος που κάνει mount το σύστημα αρχείων, για να έχει πρόσβαση στον κοινό χώρο αρχείων το πρώτο πράγμα που κάνει είναι να λάβει μια ειδική άδεια από τον κόμβο που είναι διαχειριστής του συστήματος αρχείων. Αυτή η άδεια ουσιαστικά επιτρέπει στον κόμβο να έχει πρόσβαση στον κοινό αποθηκευτικό χώρο (ακολουθώντας στη συνέχεια την γνωστή διαδικασία με τα κλειδώματα) για περιορισμένο χρονικό διάστημα. Όταν εκπνεύσει η άδεια, αν ο κόμβος θέλει να διατηρήσει την πρόσβασή του στον αποθηκευτικό χώρο θα πρέπει να ανανεώσει την άδειά του. Όταν λοιπόν ένας κόμβος καθίσταται ανενεργός δεν μπορεί να ανανεώσει την άδεια του και το GPFS επιλέγει έναν άλλο κόμβο για να κάνει recovery αν χρειαστεί.

Ενημέρωση metadata

Τα δύο συστήματα αρχείων ενημερώνουν τα metadata των αρχείων με τελείως διαφορετικό τρόπο.

Στο GFS η ενημέρωση αυτή είναι απλή. Τα metadata κάθε αρχείου ενημερώνονται από τον κόμβο που διαθέτει το κλείδωμα στο συγκεκριμένο αρχείο. Το γεγονός αυτό, όπως θα δούμε παρακάτω εξασφαλίζει την συνέπεια του συστήματος αρχείων και στις περιπτώσεις που γίνεται ανάγνωση και στις περιπτώσεις που γίνεται εγγραφή των αρχείων. Αρχικά λαμβάνουμε υπόψιν πως στο GFS δίνονται κλειδώματα για ολόκληρα αρχεία. Στην περίπτωση που ένας κόμβος θέλει να κάνει εγγραφή σε ένα αρχείο, αυτό σημαίνει πως σίγουρα θα πρέπει να ενημερώσει τα metadata του αρχείου (το μέγεθος και τον χρόνο τροποποίησης) αλλά επίσης σημαίνει πως έχει λάβει αποκλειστικό κλείδωμα για το αρχείο αυτό. Συνεπώς θα είναι και ο μόνος κόμβος που ενημερώνει τα metadata και άρα μπορεί να κάνει άφοβα την ενημέρωση.
Στην περίπτωση που θέλουν πολλοί κόμβοι να διαβάσουν το αρχείο τότε μπορούν όλοι να πάρουν ένα κλείδωμα μόνο για ανάγνωση και τα μόνα metadata που υπάρχει περίπτωση να ενημερωθούν είναι ο χρόνος πρόσβασης στο αρχείο, ο λεγόμενος atime χρόνος. Συνήθως όμως η ενημέρωση του atime χρόνου είναι απενεργοποιημένη στο GFS για να αποφευχθεί ακριβώς αυτή η περίπτωση, δηλαδή να μην μπορούν πολλοί κόμβοι να διαβάσουν ταυτόχρονα το αρχείο χωρίς να πάρουν αποκλειστικό κλείδωμα. Στις περιπτώσεις που είναι ενεργοποιημένη η ενημέρωση του atime χρόνου, συμφέρει αυτή να γίνεται σε μεγάλη χρονική περίοδο ώστε να μην προκύπτει συχνά η ανάγκη να παίρνουν οι κόμβοι αποκλειστικό κλείδωμα ενώ απλά θα διαβάσουν το αρχείο μόνο και μόνο επειδή πρέπει να ενημερωθεί ο atime χρόνος. Έτσι κι αλλιώς

ελάχιστες εφαρμογές χρησιμοποιούν τον atime χρόνο οπότε η απώλεια ενημέρωσής του έχει ελάχιστη σημασία.

Η διαχείριση των metadata είναι διαφορετική στο GPFS. Όπως είπαμε το GPFS επιτρέπει την παράλληλη πρόσβαση σε διαφορετικά μέρη του ίδιου αρχείου από διάφορους κόμβους. Αυτό σημαίνει πως δεν είναι δυνατόν κάθε κόμβος που έχει πρόσβαση σε μέρος του αρχείου να γράφει τα metadata μόνος του καθώς δεν είναι εγγυημένο πως είναι ο μόνος με πρόσβαση στο αρχείο την δεδομένη χρονική στιγμή και άρα δεν είναι εγγυημένο πως θα είναι ο μόνος που θα ενημερώνει τα metadata αυτά. Η λύση που ακολουθεί το GPFS είναι να γίνεται ενημέρωση των metadata για κάθε αρχείο από έναν κόμβο μόνο, άσχετα με το πόσοι κόμβοι έχουν ταυτόχρονη πρόσβαση στο αρχείο αυτό. Ο κόμβος που αρχικά αναλαμβάνει να ενημερώσει τα metadata ενός συγκεκριμένου αρχείου είναι ο πρώτος που θα προσπελάσει το αρχείο. Όλοι οι άλλοι κόμβοι που προσπελαύνουν ταυτόχρονα το αρχείο στέλνουν τα τροποποιημένα metadata στον αντίστοιχο κόμβο που είναι υπεύθυνος για την ενημέρωση τους. Η ταυτότητα του κόμβου που είναι υπεύθυνος για την ενημέρωση των metadata του κάθε αρχείου αποθηκεύεται στον token manager και από εκεί την μαθαίνουν και οι άλλοι κόμβοι που θέλουν να προσπελάσουν το αρχείο.
Θα ήταν παράλογο ο αρχικός κόμβος που ήταν υπεύθυνος για την ενημέρωση των metadata του αρχείου να συνεχίσει να διατηρεί αυτή την ιδιότητα και μετά το πέρας των εργασιών του με το αρχείο αυτό. Έτσι μετά το πέρας των εργασιών αυτών ο κόμβος παύει να ενημερώνει τα metadata του αρχείου και από αυτό το σημείο και πέρα ο πρώτος κόμβος που θα του ζητήσει να ενημερώσει τα συγκεκριμένα metadata θα λάβει την οδηγία να γίνει ο ίδιος υπεύθυνος για την ενημέρωσή τους. Η αλλαγή του κόμβου που χειρίζεται τα metadata συνεπάγεται θα πρέπει να ενημερωθεί και ο token manager του συστήματος αρχείων για την αλλαγή ώστε οι επόμενοι κόμβοι που θα προσπελάσουν το αρχείο να γνωρίζουν την ταυτότητα του νέου υπεύθυνου για την ενημέρωση των metadata.

Ύπαρξη του Data Shipping Mode στο GPFS

Μια δυνατότητα που υπάρχει στο GPFS και όχι στο GFS είναι το λεγόμενο Data Shipping Mode. Το Data Shipping Mode βοηθάει να αυξάνεται ο ρυθμός διαμεταγωγής των δεδομένων προς τους κόμβους όταν αυτοί τρέχουν εφαρμογές κάνουν πολλές προσπελάσεις σε μικρά τμήματα δεδομένων (μικρότερα του ενός block δεδομένων) και επιπλέον δεν απαιτούν να ακολουθείται η POSIX σημασιολογία.
Ο τρόπος με τον οποίο λειτουργεί είναι ο εξής:
Τα blocks των αρχείων ανατίθενται στους κόμβους που συμμετέχουν στο σύστημα αρχείων με round robin αλγόριθμο ώστε κάθε block να διαβάζεται και να γράφεται μόνο από έναν κόμβο. Έτσι κάθε αίτηση που περιλαμβάνει δεδομένα που είναι αποθηκευμένα σε ένα block προωθείται από τον κόμβο από τον οποίο δημιουργήθηκε, στον κόμβο που είναι υπεύθυνος για το συγκεκριμένο block. Με αυτό τον τρόπο αν ζητούνται συνεχόμενα blocks ενός αρχείου από έναν κόμβο μπορούν να γίνουν παράλληλες αιτήσεις στους κόμβους που είναι υπεύθυνοι για τα συγκεκριμένα block.
Είναι προφανές ότι ο τρόπος αυτός πρόσβασης στα κοινά δεδομένα εξαλείφει την ανάγκη ύπαρξης κλειδωμάτων αφού οι αιτήσεις για πρόσβαση στα δεδομένα σειριοποιούνται. Το αρνητικό γεγονός που συνεπάγεται το Data Shipping Mode είναι αύξηση της δικτυακής κίνησης πάνω στο δίκτυο διασύνδεσης, με μηνύματα και

δεδομένα που αφορούν όχι δικτυακές επικοινωνία μεταξύ των εφαρμογών που τρέχουν στους κόμβους του cluster, αλλά ανάγκες του συστήματος αρχείων. Σε περιπτώσεις που γίνεται κοινή χρήση πολύ μικρών κομματιών των αρχείων ο τρόπος αυτός πρόσβασης στα δεδομένα είναι αποδοτικότερος καθώς εξαλείφει την ανάγκη ανταλλαγής πολλών μηνυμάτων που αφορούν τα κλειδώματα και επιπλέον δεν χάνεται χρόνος για την ακύρωση δεδομένων από τις cache και εγγραφής αυτών στο αποθηκευτικό μέσο.

Σημειώνουμε ότι η ενεργοποίηση του Data Shipping Mode πραγματοποιείται προγραμματιστικά από κάθε εφαρμογή που θέλει να κάνει χρήση της συγκεκριμένης δυνατότητας. Η ενεργοποίηση αυτή γίνεται ξεχωριστά για κάθε αρχείο στο οποίο θα υπάρξει κοινή πρόσβαση από τους κόμβους που τρέχουν την εφαρμογή.

Πραγματικά αποτελέσματα για τον ρυθμό διαμεταγωγής που επιτυγχάνεται με το Data Shipping Mode ενεργοποιημένο και σύγκριση με τον κανονικό τρόπο πρόσβασης στα δεδομένα για I/O αιτήσεις διαφόρων μεγεθών παρουσιάζονται στο GPFS: A Shared Disk File System for Large Computing Clusters. Εκεί βλέπουμε ότι όταν υπάρχει κοινή πρόσβαση σε δεδομένα μικρότερα του ενός block ενός αρχείου (το ελάχιστο ποσό δεδομένων για το οποίο μπορεί να δοθεί κλείδωμα σε κανονική λειτουργία), το Data Shipping Mode υπερέχει ξεκάθαρα. Σε περιπτώσεις όμως που η κοινή χρήση δεδομένων γίνεται σε μεγάλα τμήματα του αρχείου, ο κανονικός τρόπος λειτουργίας με τα κλειδώματα έχει ως αποτέλεσμα μεγαλύτερους ρυθμούς διαμεταγωγής.

Όπως περιγράφεται στο κεφάλαιο 8, μπορούμε να δούμε το Data Shipping Mode να δουλεύει στην πράξη μέσω καταγραφών. Εκεί θα χρησιμοποιήσουμε μια εφαρμογή που έχει την δυνατότητα να χρησιμοποιήσει Data Shipping Mode, για να δούμε ότι όταν αυτό ενεργοποιείται, ακόμα και αν διαφορετικοί κόμβοι προσπελαύνουν τα ίδια δεδομένα σε ένα αρχείο, δεν υπάρχουν στις καταγραφές δύο κόμβοι που προσπελαύνουν τα ίδια block.

Ύπαρξη subblocks στο GPFS

Αν και τα δύο συστήματα αρχείων υποστηρίζουν διάφορα μεγέθη block δεδομένων που μπορούν να επιλεχθούν από τον δημιουργό του συστήματος αρχείων ανάλογα με τις ανάγκες αποθήκευσης, εντούτοις το GPFS υποστηρίζει μια επιπλέον λειτουργία που μειώνει την σπατάλη του αποθηκευτικού χώρου.
Συγκεκριμένα υποστηρίζει τον χωρισμό ενός block σε 32 ισομεγέθη subblocks στα οποία μπορεί ν' αποθηκεύσει μικρά αρχεία ή το τελευταίο μέρος των δεδομένων μεγάλων αρχείων. Με αυτό τον τρόπο εξοικονομείται αποθηκευτικός χώρος καθώς τα μικρά αρχεία καταλαμβάνουν χώρο μόνο όσα τριακοστά δεύτερα ενός block απαιτούνται αντί για ένα ολόκληρο block.
Η εξοικονόμηση αποθηκευτικού χώρου με αυτό τον τρόπο έρχεται όμως με ένα τίμημα. Το τίμημα αυτό είναι η αύξηση των συγκρούσεων για την απόκτηση κλειδωμάτων. Αυτό μπορούμε να το καταλάβουμε αν σκεφτούμε το ακόλουθο σενάριο:
Σε ένα μόνο block αποθηκεύονται κάποια μικρά αρχεία. Επίσης έχουμε ορισμένους κόμβους που ο καθένας θέλει να προσπελάσει ένα μόνο από τα αρχεία αυτά με τον καθένα να θέλει πρόσβαση σε διαφορετικό αρχείο. Υποθέτουμε επίσης ότι τυχαίνει

όλοι οι κόμβοι να θέλουν την ίδια στιγμή τα αρχεία. Αν τα μικρά αρχεία είχαν αποθηκευτεί σε ένα block το καθένα, θα είχε χαθεί πολύς αποθηκευτικός χώρος, αλλά θα μπορούσαν όλοι οι κόμβοι να προσπελάσουν ταυτόχρονα τα αρχεία και να μην συγκρούονται για τα κλειδώματα. Αντίθετα στην περίπτωση που τα μικρά αρχεία αποθηκεύονται όλα σε ένα block, επειδή στο GPFS το μικρότερο δυνατό κλείδωμα που μπορεί να δοθεί αντιστοιχεί σε ένα block δεδομένων, θα υπάρχει σύγκρουση όλων των κόμβων για το ίδιο κλείδωμα άσχετα με το γεγονός ότι ζητάνε πρόσβαση σε διαφορετικά αρχεία.

Επίσης πρέπει να σημειωθεί πως άσχετα με την ύπαρξη των subblocks, το μικρότερο μέγεθος δεδομένων που μπορεί να μεταφερθεί από τον αποθηκευτικό χώρο στο μέρος της μνήμης που διαχειρίζεται το GPFS παραμένει ένα block. Απλά όταν ζητάτε ένα subblock από μια εφαρμογή, το GPFS θα αντιγράψει από την μνήμη που διαχειρίζεται στο κομμάτι μνήμης εφαρμογής, μόνο το subblock.
Η μέθοδος αυτή δεν χρησιμοποιείται μόνο για τα subblocks και μόνο από το GPFS, αλλά για οποιοδήποτε μέγεθος δεδομένων είναι μικρότερο του ενός block και από κάθε σύστημα αρχείων. Όπως είπαμε και πριν, όταν είναι γνωστό πως η πρόσβαση στα αρχεία θα είναι κατά κόρον μικρές σε δεδομένα(μικρότερες του ενός block) και δεν απαιτείται η ύπαρξη της POSIX σημασιολογίας από την εφαρμογή που τρέχει, είναι αποδοτικότερη η χρήση του Data Shipping Mode.

Ύπαρξη επιπλέον προγραμματιστικού interface στο GPFS

Όλα τα συστήματα αρχείων στο Linux, υλοποιούν τις γνωστές κλήσεις συστήματος που αφορούν την πρόσβαση στο σύστημα αρχείων δίνοντας την δυνατότητα σε εφαρμογές που έχουν γραφτεί για περιβάλλον UNIX να τρέχουν απρόσκοπτα όταν τα χρησιμοποιούν.
Το GPFS όμως παρέχει ένα επιπλέον προγραμματιστικό inteface που δίνει την δυνατότητα σε εφαρμογές που είναι προορισμένες να λειτουργούν με το GPFS να μπορούν να αλληλεπιδράσουν καλύτερα με αυτό και να επιτύχουν ακόμα υψηλότερες I/O επιδόσεις. Ένας από τους λόγους που μπορούν να αυξηθούν οι I/O επιδόσεις με χρήση του προγραμματιστικού interface του GPFS, είναι η δυνατότητα η εφαρμογή να αναφέρει στο GPFS ότι μελλοντικά θα διαβάσει κάποια δεδομένα του συστήματος αρχείων. Με αυτό τον τρόπο το GPFS μπορεί να φέρει νωρίς τα δεδομένα αυτά και να τα βάλει στην cache του κόμβου έτσι ώστε όταν τελικά γίνει η πραγματική ανάγνωση τα δεδομένα να διαβαστούν από εκεί παρακάμπτοντας το σύστημα αποθήκευσης.

Διαμοιρασμός συστήματος αρχείων σε πολλά clusters

Το GPFS δίνει την δυνατότητα σε διαφορετικά GPFS cluster να μοιράζονται τα συστήματα αρχείων που διαθέτουν. Δηλαδή αν έχουμε δύο cluster που έχουν δικτυακή σύνδεση μεταξύ τους, μπορεί το ένα cluster να χρησιμοποιεί τα συστήματα αρχείων που έχουν στηθεί στον αποθηκευτικό χώρο του άλλου, σαν να ήταν ο χώρος αυτός μέρος του. Η δυνατότητα αυτή λείπει ολοκληρωτικά από το GFS.

Τρόπος δέσμευσης ελεύθερου χώρου

Τα δύο συστήματα αρχείων διαφοροποιούνται και στον τρόπο που δεσμεύουν αποθηκευτικό χώρο. Όπως γνωρίζουμε κάθε σύστημα αρχείων δημιουργεί στις εφαρμογές την αφαίρεση ότι τα αρχεία αποτελούνται από συνεχόμενα λογικά block, ενώ το ίδιο αναλαμβάνει την μετατροπή των λογικών αυτών block σε φυσικά block του αποθηκευτικού χώρου.

Στο GPFS κάθε κόμβος του συστήματος αρχείων διαχειρίζεται την δέσμευση και την αποδέσμευση ενός συνεχόμενου μέρους των blocks του αποθηκευτικού χώρου. Τα blocks αυτά δεν είναι αποθηκεύονται συνεχόμενα σε ένα δίσκο, αλλά μοιράζονται σε όλο τον αποθηκευτικό χώρο αν είναι δυνατόν, για να μπορεί να γίνει καλύτερη εκμετάλλευση του striping. Το GPFS κάνει προσπάθεια να εκχωρεί την διαχείριση του αποθηκευτικού χώρου με τέτοιο τρόπο ώστε κάθε κόμβος να χρειάζεται να δεσμεύει και να αποδεσμεύει μόνο τον χώρο για τον οποίο είναι υπεύθυνος. Στις περιπτώσεις που αυτό δεν είναι δυνατό, οι αιτήσεις δέσμευσης και αποδέσμευσης χώρου που διαχειρίζονται άλλοι κόμβοι, προωθούνται και εξυπηρετούνται από αυτούς. Οι πληροφορίες που κρατά κάθε κόμβος για τον χώρο για τον οποίο είναι υπεύθυνος αφορούν για κάθε block αν είναι δεσμευμένο ή όχι καθώς επίσης και subblock του κάθε block αν είναι δεσμευμένο ή όχι.

Στο GFS κάθε κόμβος είναι υπεύθυνος για την δέσμευση και αποδέσμευση ορισμένων συνεχόμενων φυσικών blocks του αποθηκευτικού χώρου. Η λογική πίσω από αυτή την απόφαση είναι ότι αφού το GFS από μόνο του δεν κάνει striping, το γεγονός ότι δεδομένα που αντιστοιχούν σε ίδια αρχεία βρίσκονται σε διαδοχικά blocks εξυπηρετεί την πιο γρήγορη πρόσβαση σε αυτά. Η λογική αυτή είναι η ίδια με την λογική που ακολουθούν και τα συστήματα αρχείων που χρησιμοποιούνται σε τοπικά συνδεδεμένα μέσα αποθήκευσης, όπου προσπαθούν να αποφύγουν τον κατακερματισμό των αρχείων αποθηκεύοντας τα σε γειτονικά τμήματα του μέσου αποθήκευσης. Κάθε κόμβος κρατά πληροφορίες μόνο για τμήματα ελεύθερου αποθηκευτικού χώρου που βρίσκονται εντός του μέρους του αποθηκευτικού χώρου για το οποίο είναι υπεύθυνος. Για κάθε τμήμα ελεύθερο χώρου, αρκεί ο υπεύθυνος κόμβος να κρατά το πρώτο ελεύθερο block και το πλήθος τους. Τα δεσμευμένα blocks για τα οποία είναι υπεύθυνος κάθε κόμβος δεν κρατούνται επειδή προκύπτουν εύκολα από την αφαίρεση του ελεύθερου χώρου από τον συνολικό χώρο για τον οποίο είναι υπεύθυνος ο κόμβος.

Ύπαρξη του GPFS Tracing

Στο GPFS υπάρχει η δυνατότητα καταγραφής πολλών γεγονότων που αφορούν τις διάφορες ενέργειες που κάνει. Μεταξύ των ενεργειών που μπορούν να καταγραφούν, είναι προσβάσεις στον αποθηκευτικό χώρο, διαχείριση κλειδωμάτων και αλληλεπίδραση με το VFS του πυρήνα του Linux. Σύμφωνα με την κατασκευάστρια εταιρεία, ο κύριος λόγος ύπαρξης του συστήματος καταγραφής, είναι εξεύρεση της πηγής διαφόρων προβλημάτων που μπορεί να προκύψουν. Το σύστημα αυτό λείπει εξ' ολοκλήρου από το GFS.

Στο κεφάλαιο 8 χρησιμοποιείται εκτός των άλλων και το σύστημα καταγραφής του GPFS προκειμένου να εξαχθούν ορισμένα χρήσιμα σενάρια για την λειτουργία του.

Άδεια Χρήσης

Μια σημαντική μεν, μη τεχνική διαφορά δε, είναι ότι το GFS αποτελεί ελεύθερο λογισμικού που εκδίδεται κάτω από την άδεια GPL, ενώ το GPFS εκδίδεται κάτω από εμπορική άδεια χρήσης με υψηλό κόστος. Παρ' όλα αυτά η IBM δίνει την δυνατότητα σε πανεπιστήμια να αποκτήσουν το GPFS χωρίς κόστος με χρήση ειδικής άδειας για ερευνητικούς σκοπούς.

4.9 Ομοιότητες GFS και GPFS

Τα δύο αυτά συστήματα αρχείων πέραν των διαφορών που παρουσιάζουν, έχουν και πολλά κοινά στοιχεία. Τα στοιχεία αυτά παραθέτουμε αμέσως τώρα.

Δυνατότητα απευθείας και NBD πρόσβασης στον αποθηκευτικό χώρο

Μπορούν και τα δύο υπό μελέτη συστήματα αρχείων να δημιουργηθούν πάνω σε δίκτυο SAN χωρίς την ύπαρξη servers για την διεκπαιρέωση της I/O κίνησης αν όλοι οι κόμβοι του cluster έχουν απευθείας σύνδεση στο SAN, αλλά και σε configuration όπου υπάρχουν servers που εξυπηρετούν αιτήσεις από κόμβους του cluster που δεν έχουν απευθείας πρόσβαση στον αποθηκευτικό χώρο.
Για την επίτευξη της μεταφοράς δεδομένων σε κόμβους που δεν έχουν απευθείας πρόσβαση στον αποθηκευτικό χώρο, η αρχιτεκτονική σχεδίαση που ακολουθείται είναι αυτή του Network Block Device (NBD) που υπάρχει εδώ και καιρό στο λειτουργικό σύστημα Linux.
Σε αυτήν, οι servers που έχουν απευθείας πρόσβαση στον αποθηκευτικό χώρο κάνουν γνωστό το γεγονός αυτό στους άλλους κόμβους. Οι κόμβοι που δεν έχουν άμεση πρόσβαση στον αποθηκευτικό χώρο ονομάζονται clients. Χρησιμοποιώντας ένα kernel module οι clients δημιουργούν ένα εικονικό block device προς χρησιμοποίηση από το σύστημα αρχείων και έχουν πρόσβαση στον αποθηκευτικό χώρο μέσω των servers.

Στο κεφάλαιο 6 περιγράφεται εκτενώς η διαδικασία εγκατάστασης των δύο συστημάτων αρχείων. Εκεί βλέπουμε στην πράξη ότι και τα δύο μπορούν να στηθούν σε SAN configuration και σε NBD configuration.

Μηχανισμοί Αποκατάστασης από σφάλματα (Recovery)

Οι μηχανισμοί recovery που χρησιμοποιεί το GPFS είναι παρόμοιοι με τους αντίστοιχους του GFS και βασίζονται στην χρήση logs. Σε γενικές γραμμές η λειτουργία του συστήματος recovery είναι η εξής:
Κάθε κόμβος που χρησιμοποιεί το σύστημα αρχείων γράφει σε ένα ξεχωριστό log τις ενέργειές του που επιφέρουν τροποποίηση των metadata. Δηλαδή για κάθε τέτοια ενέργεια που πρόκειται να κάνει ένας κόμβος, γράφονται πρώτα τα απαραίτητα στοιχεία στο αντίστοιχο log και μετά εκτελείται η ενέργεια. Τα logs ενημερώνονται κατευθείαν στον δίσκο και όχι στην μνήμη των κόμβων για να είναι πάντα ενημερωμένα και να μην υπάρχει πιθανότητα απώλειάς τους.

Και τα δύο συστήματα αρχείων καταγράφουν στα logs τους, μόνο ενέργειες που αφορούν metadata και όχι ενέργειες που αφορούν τα δεδομένα των αρχείων. Τα logs αυτά βρίσκονται στον κοινό χώρο αποθήκευσης και άρα όταν παραστεί ανάγκη, είναι προσβάσιμα από όλους τους κόμβους που χρησιμοποιούν το ίδιο σύστημα αρχείων. Έτσι λοιπόν όταν ένας κόμβος που χρησιμοποιεί ένα σύστημα αρχείων τεθεί εκτός λειτουργίας, κάποιος άλλος κόμβος που χρησιμοποιεί το ίδιο σύστημα αρχείων αναλαμβάνει να το αποκαταστήσει βασισμένο στο log του ανενεργού κόμβου.

Χρήση πολλών συστημάτων αρχείων από ένα cluster

Το GFS και το GPFS επιτρέπουν να χρησιμοποιούνται πολλά διαφορετικά συστήματα αρχείων ταυτόχρονα από τους κόμβους του cluster.
Δηλαδή οι κόμβοι του cluster μπορούν να συμμετέχουν ταυτόχρονα σε πολλά GFS ή GPFS συστήματα αρχείων χωρίς κανένα περιορισμό. Αυτό βοηθάει ιδιαίτερα όταν τα συστήματα αρχείων γίνονται πολύ μεγάλα.

Δυνατότητα για NFS export

Τα δύο συστήματα αρχείων που μελετάμε, μπορούν να γίνουν exported από το cluster ως NFS συστήματα αρχείων.
Η δυνατότητα αυτή επιτρέπει και σε κόμβους που δεν βρίσκονται στο cluster να μοιράζονται το σύστημα αρχείων. Το GPFS επιπλέον επιτρέπει την εξαγωγή ως CIFS σύστημα αρχείων.

Τρόπος αναζήτησης μέσω σε directory

Η τεχνική του Επεκτάσιμου Hashing (Extendible Hashing) χρησιμοποιείται από το GPFS καθώς και από το GFS προκειμένου να πραγματοποιείται γρήγορα η αναζήτηση μέσα σε καταλόγους που περιλαμβάνουν πάρα πολλά αρχεία.

Το σύστημα αρχείων, όπως ξέρουμε δημιουργεί την αφαίρεση ότι ο αποθηκευτικός χώρος αποτελείται από έναν πίνακα από blocks δεδομένων. Συνεπώς κάθε block είναι αριθμημένο. Όταν αναζητούμε ένα αρχείο με βάση το όνομα μέσα σε έναν κατάλογο, ο σκοπός μας είναι να βρούμε το block δεδομένων που περιέχει την dentry (directory entry) δομή του αρχείου, καθώς αυτή περιέχει το όνομα του αρχείου και έχει έναν δείκτη προς την inode δομή του αρχείου.

Σε κάθε dentry δομή που αναπαριστά κατάλογο, υπάρχει ένας δείκτης προς τις dentry δομές των αρχείων που ανήκουν σε αυτό τον κατάλογο. Η συνήθης οργάνωση των dentry αυτών δομών είναι σε μια μη ταξινομημένη λίστα. Η οργάνωση σε μη ταξινομημένη λίστα είναι επαρκής στην περίπτωση που ο κατάλογος δεν περιέχει πολλά αρχεία αλλά κρίνεται ανεπαρκής όταν θέλουμε να αναζητήσουμε ένα αρχείο μέσα σε ένα κατάλογο που περιέχει πάρα πολλά αρχεία. Αυτό διότι οι dentry δομές καταλαμβάνουν πολλά blocks δεδομένων και για την εύρεση της κατάλληλης dentry δομής ενδέχεται να χρειαστεί να διατρεχθεί όλη η λίστα, άρα να γίνουν πολλές προσβάσεις στον δίσκο.

Αντίθετα, με την τεχνική του Επεκτάσιμου Hashing η αναζήτηση μπορεί να πραγματοποιηθεί σε πολύ μικρότερο χρόνο καθώς στην χειρότερη περίπτωση αρκεί να ελεγχθούν όλες οι dentry δομές που βρίσκονται σε ένα block δεδομένων. Αυτό γιατί το block στο οποίο αποθηκεύεται κάθε dentry δομή μπορεί να βρεθεί απευθείας με την χρησιμοποίηση μιας συνάρτησης που λαμβάνει ως όρισμα το όνομα του αρχείου. Σε γενικές γραμμές η τεχνική λειτουργεί ως εξής:
Στο επεκτάσιμο hashing ο αριθμός του block δεδομένων μέσα στο οποίο βρίσκεται η dentry δομή που αντιστοιχεί σε κάθε αρχείο, μπορεί να βρεθεί αν από την εφαρμογή μιας hash συνάρτηση πάνω στο όνομα του αρχείου κρατήσουμε τα n δεξιά bits της τιμής που επιστρέφει. Ο αριθμός n εξαρτάται από το πλήθος των αρχείων που βρίσκονται στον κατάλογο στον οποίο γίνεται η αναζήτηση. Η αναζήτηση λοιπόν μπορεί να γίνει πολύ γρήγορα. Το άλλο θέμα που πρέπει να επιλυθεί είναι ότι δεν θέλουμε να δεσμεύσουμε εξ αρχής πολλά block δεδομένων για την αποθήκευση των dentry δομών, αλλά η δέσμευση να γίνεται δυναμικά ανάλογα με τον αριθμό των αρχείων του καταλόγου.
Η αντιμετώπιση αυτού του προβλήματος με την τεχνική του Επεκτάσιμου Hashing γίνεται ως εξής: κατά τη δημιουργία νέου αρχείου, όταν δεν υπάρχει χώρος για αποθήκευση της νέας dentry δομής στο block που υποδεικνύεται από την συνάρτηση hash, το block αυτό διαχωρίζεται σε δύο. Ο αριθμός του νέου block που προκύπτει είναι ο αριθμός του παλαιού block με την αντικατάσταση του bit n+1 που ήταν μηδέν, με ένα. Στη συνέχεια τα αρχεία του παλαιού block που είχαν τιμή 1 στο n+1 bit της hash τιμής τους, μεταφέρονται στο νέο block. Όλα τ' άλλα blocks που περιέχουν dentry δομές για τον συγκεκριμένο κατάλογο μένουν ανέπαφα. Με αυτό τον τρόπο δεσμεύονται όσα blocks δεδομένων είναι απαραίτητα για την αποθήκευση των dentry δομών.

Κεφάλαιο 5.
Τύποι Παράλληλων Υπολογιστικών Συστημάτων

5.1 Εισαγωγή

Στο κεφάλαιο αυτό περιγράφονται οι βασικοί τύποι συστημάτων παράλληλης επεξεργασίας, κατηγοριοποιημένοι σύμφωνα με την υποστήριξη της παραλληλίας που προσφέρει το hardware. Η περιγραφή αυτή γίνεται καθώς η μελέτη μας έχει σχέση με παράλληλα συστήματα, αλλά δεν εμβαθύνει πολύ καθώς αντικείμενο μας είναι τα παράλληλα και συστήματα αρχείων κοινού μέσου, ήτοι το παράλληλο I/O και όχι η παράλληλη επεξεργασία.
Μεγαλύτερη βαρύτητα δίνεται στην περιγραφή των clusters καθώς είναι ο τύπος παράλληλου συστήματος που χρησιμοποιήθηκε στη μελέτη αυτή.

5.2 Multicore Systems (Πολυπύρηνα Συστήματα)

Τα πολυπύρηνα συστήματα παράλληλης επεξεργασίας είναι ουσιαστικά απλοί υπολογιστές που βασίζονται στην χρήση ενός πολυπύρηνου επεξεργαστή. Ο πολύπύρηνος επεξεργαστής διαθέτει περισσότερες πολλαπλές μονάδες εκτέλεσης εντολών (δηλαδή πυρήνες), με αποτέλεσμα να μπορεί να εκτελέσει ταυτόχρονα τόσες εντολές όσοι και οι πυρήνες που διαθέτει. Κάθε πυρήνας μπορεί να διαθέτει όλες τις γνωστές τεχνολογίες επεξεργαστών, όπως pipelining, superscaling, out of order execution κτλ.
Οι πολυπύρηνος επεξεργαστές διαθέτουν επίσης ενσωματωμένες cache, που μπορούν να μοιράζονται ανάμεσα στους πυρήνες ή να προορίζονται για αποκλειστική χρήση από ένα πυρήνα.
Οι πιο γνωστοί πολυπύρηνοι επεξεργαστές είναι ο Cell της IBM και η σειρά Core της Intel. Επίσης οι επεξεργαστές που χρησιμοποιούν οι κάρτες γραφικών (GPU), ανήκουν στην κατηγορία των πολυπύρηνων επεξεργαστών.

5.3 Symmetric Multiprocessing Systems (Συμμετρικά Συστήματα Πολυεπεξεργασίας)

Τα συστήματα αυτά είναι υπολογιστές που διαθέτουν περισσότερους από έναν αυτόνομους επεξεργαστές. Οι επεξεργαστές αυτοί συνδέονται μεταξύ τους μέσω ενός διαδρόμου και έχουν όλοι άμεσοι πρόσβαση στην κεντρική μνήμη του συστήματος. Μόνο ένας επεξεργαστής μπορεί να έχει πρόσβαση στην κεντρική μνήμη σε κάθε χρονική στιγμή, αφού όλοι οι επεξεργαστές χρησιμοποιούν τον ίδιο διάδρομο για να την προσπελάσουν. Το γεγονός αυτό αποτελεί τον κύριο ανασταλτικό παράγοντα για το scalling αυτών των συστημάτων. Στην πράξη σπάνια τέτοια συστήματα διαθέτουν περισσότερους από 32 επεξεργαστές.
Για να είναι δυνατή η χρήση πολλών επεξεργαστών σε ένα τέτοιο σύστημα, εκτός από την υποστήριξη της μητρικής κάρτας, απαιτείται και υποστήριξη από τους ίδιους τους επεξεργαστές.
Ως αυτόνομοι επεξεργαστές μπορούν να χρησιμοποιηθούν πολυπύρηνοι επεξερ-

γαστές αν βεβαίως υποστηρίζουν αυτή την δυνατότητα, αυξάνοντας ακόμα περισσότερο την παραλληλία.

5.4 Massively Parallel Processing(MPP) Systems

Αυτά τα συστήματα αποτελούνται από έναν μοναδικό υπερυπολογιστή που περιέχει εκατοντάδες ή χιλιάδες επεξεργαστές. Η διασύνδεση των επεξεργαστών μπορεί να γίνει με διάφορα σχήματα που διαφέρουν σε επιδόσεις, δυνατότητα επέκτασης και κόστος. Κάθε επεξεργαστής προσπελαύνει μνήμη που είναι αποκλειστικά δική του και διαθέτει δικό του αντίγραφο του λειτουργικού συστήματος.
Σύμφωνα με την λίστα που εξέδωσε το top500.org τον Ιούνιο του 2008, το 19.6% των πιο δυνατών υπερυπολογιστών του κόσμου ακολουθούν την αρχιτεκτονική MPP. Μεταξύ αυτών και ο δεύτερος πιο γρήγορος υπερυπολογιστής του κόσμου, ο Blue Gene/L που βρίσκεται στο Lawrence Livermore National Laboratory στις Η.Π.Α.

5.5 Συστοιχίες Υπολογιστών (Computer Clusters)

Τα Computer Clusters ή απλά clusters είναι ένα σύνολο από υπολογιστικά συστήματα, μονάδες αποθήκευσης και δίκτυα διασύνδεσης που δουλεύουν μαζί στενά, σχηματίζοντας έτσι ένα συνολικό σύστημα που στους χρήστες και στους υπολογιστές του εξωτερικού περιβάλλοντος, φαίνεται ως ένα ενιαίο σύστημα υψηλής απόδοσης και/ή διαθεσιμότητας.
Συνήθως τα cluster έχουν πολύ μικρότερο κόστος από μεμονωμένα συστήματα που παρέχουν παραπλήσιες επιδόσεις και/ή διαθεσιμότητα.
Τα clusters ταξινομούνται στις παρακάτω κατηγορίες:

Υψηλής Διαθεσιμότητας (High Availability)

Ο σκοπός αυτού του είδους cluster είναι να παρέχει ορισμένες υπηρεσίες αδιάκοπα παίρνοντας μέτρα για να προφυλαχθεί από την απρόσκοπτη διακοπή λειτουργίας ορισμένων συστημάτων του. Ο σκοπός αυτός επιτυγχάνεται με την χρήση επιπλέον hardware που αναλαμβάνει τις λειτουργίες άλλου hardware όταν αυτό τεθεί εκτός λειτουργίας και γενικά με την εξάλειψη των single point of failure. Το ελάχιστο σε μέγεθος τέτοιο cluster αλλά και συνάμα το πιο κοινό είναι το cluster δύο κόμβων.

Κατανομής Φορτίου(Load Balancing)

Αυτό το είδος cluster χρησιμοποιείται προκειμένου να κατανεμηθεί ισόποσα το υπολογιστικό φορτίο που δέχεται το cluster από το εξωτερικό του περιβάλλον. Συνήθως σε αυτά τα cluster υπάρχουν ορισμένοι κόμβοι που αναλαμβάνουν πρώτον να δέχονται το φορτίο από το εξωτερικό περιβάλλον(και οι οποίο μπορούν να αντικαταστήσουν ο ένας τον άλλο ώστε το cluster να είναι πάντα ενεργό), δεύτερον να κατανέμουν το φορτίο σε κόμβους του cluster που είναι επιφορτισμένοι με την επεξεργασία του φορτίου και τρίτον να επιστρέφουν στο εξωτερικό περιβάλλον τα αποτελέσματα της επεξεργασίας αυτής.

Υψηλών Υπολογιστικών Επιδόσεων(High Perfomance Computing)

Σε αυτά τα cluster όλοι οι κόμβοι αναλαμβάνουν να εκτελέσουν κάποιες, συνήθως παράλληλες και ιδιαίτερα απαιτητικές σε υπολογιστική ισχύ, εφαρμογές.
Μέσω της παραλληλοποίησης των εφαρμογών καθώς και της προσεκτικής επιλογής υποσυστημάτων που μπορούν να αποτελέσουν σημείο συμφόρησης όπως για παράδειγμα το δίκτυο διασύνδεσης ή το σύστημα αρχείων, τα cluster αυτά να επιτύχουν πολύ υψηλές επιδόσεις.

Για να πειστεί κανείς για την υπολογιστική ισχύ που μπορούν να έχουν αυτά τα cluster, αρκεί να αναφερθεί ότι ο Roadrunner, ο πιο δυνατός υπερυπολογιστής του κόσμου σύμφωνα με την λίστα που εξέδωσε το top500.org τον Ιούνιο του 2008 ακολουθεί την αρχιτεκτονική του cluster Υψηλών Υπολογιστικών Επιδόσεων.
Το ίδιο άλλωστε συμβαίνει και για το 80% των συστημάτων της λίστας αυτής.
Τα cluster αυτά συνήθως αυξάνουν τις επιδόσεις τους με την προσθήκη νέων κόμβων, δηλαδή γίνονται scale καλά.

Οι εφαρμογές που εκμεταλλεύονται τα πλεονεκτήματα που προσφέρουν τα clusters είναι κυρίως οι επιστημονικές εφαρμογές, οι εφαρμογές επεξεργασίας γραφικών, οι εφαρμογές που χρησιμοποιούνται στην παραγωγή ταινιών καθώς και οι εμπορικοί εξυπηρετητές, όπως εξυπηρετητές ιστού, ηλεκτρονικού ταχυδρομείου και βάσεων δεδομένων.
Οι δύο πρώτες κατηγορίες χρησιμοποιούν την μεγάλη υπολογιστική δύναμη που παρέχουν τα clusters, ενώ η τελευταία κατηγορία εκμεταλλεύεται την υψηλή διαθεσιμότητα και την κατανομή φορτίου που παρέχουν.
Πιο συγκεκριμένα οι επιστημονικές εφαρμογές αφορούν την μελέτη περιβαλλοντικών φαινομένων, πυρηνικών δοκιμών, βιολογικών, φαρμακευτικών και ιατρικών διαδικασιών, αμυντικών συστημάτων και οικονομικών μοντέλων.

Λογισμικό που παρέχει clustering είναι διαθέσιμο για μια ποικιλία λειτουργικών συστημάτων όπως Windows, Mac OS X, Solaris κτλ. Τα πλεονεκτήματα όμως της χρήσης του λειτουργικού συστήματος Linux το καθιστούν μια εξαιρετική επιλογή για την ανάπτυξη clusters στο περιβάλλον του.
Τα πλεονεκτήματα αυτά είναι:

- Μεγάλο εύρος υποστηριζόμενου hardware
- Εξαιρετική σταθερότητα
- Διαθέσιμος πηγαίος κώδικας
- Ύπαρξη ελάχιστων ιών
- Ύπαρξη πλήθους δωρεάν εργαλείων και εφαρμογών

Στο περιβάλλον του Linux ένα πλήθος διαφόρων τεχνολογιών χρησιμοποιούνται για την ανάπτυξη και εκμετάλλευση των clusters. Μερικές από αυτές είναι οι παρακάτω:

- Το Beowulf είναι μια κατηγορία cluster που τρέχουν (και) σε Linux και παρέχουν ένα cluster Υψηλών Υπολογιστικών Επιδόσεων χρησιμοποιώντας απλούς καθημερινούς υπολογιστές για να τρέχουν παράλληλες εφαρμογές.
- Το MPI είναι μια τυποποίηση που στόχο έχει να επιτρέπει την ανάπτυξη παράλληλων προγραμμάτων που θα τρέχουν σε συστήματα κατανεμημένης μνήμης όπως τα clusters. Το MPI από μόνο του αποτελεί μια τυποποίηση και

δεν είναι συνδεδεμένο με μια συγκεκριμένη υλοποίηση. Κάθε υλοποίηση πρέπει να δημιουργεί την βιβλιοθήκη MPI ακολουθώντας πιστά την τυποποίηση.
Επίσης η τυποποίηση του MPI δεν είναι συνδεδεμένη με κάποια συγκεκριμένη γλώσσα προγραμματισμού, συνεπώς κάθε υλοποίηση μπορεί να δημιουργεί την απαραίτητη βιβλιοθήκη σε όποια γλώσσα επιθυμεί. Επίσης η γλώσσα προγραμματισμού που χρησιμοποιείται για την υλοποίηση της βιβλιοθήκης MPI είναι ανεξάρτητη από τις γλώσσες προγραμματισμού που χρησιμοποιούνται για την ανάπτυξη εφαρμογών που θα χρησιμοποιούν την βιβλιοθήκη. Οι περισσότερες υλοποιήσεις του MPI επιτρέπουν την ανάπτυξη παράλληλων προγραμμάτων σε C, C++ και Fortran. Οι πιο γνωστές υλοποιήσεις του MPI είναι η MPICH και η Open MPI. Πρέπει επίσης ν' αναφέρουμε πως αν και η τυποποίηση του MPI αναφέρει πως χρησιμοποιεί ανταλλαγή μηνυμάτων σε περιβάλλον κατανεμημένης μνήμης για επικοινωνία μεταξύ των κόμβων, εντούτοις τίποτα δεν εμποδίζει την ανάπτυξη της βιβλιοθήκης σε περιβάλλον κοινής μνήμης χωρίς τη χρήση μηνυμάτων, αρκεί να ακολουθείται πιστά η σημασιολογία της τυποποίησης. Πράγματι υπάρχουν διαθέσιμες και τέτοιες υλοποιήσεις.

- Το Linux HA (High Availability Linux) είναι ένα λογισμικό πακέτο που επιτρέπει την δημιουργία cluster Υψηλής Διαθεσιμότητας.
- Το OSCAR είναι ένα πρόγραμμα για Linux που επιτρέπει στους χρήστες να δημιουργούν εύκολα ένα cluster Υψηλών Υπολογιστικών Επιδόσεων, τύπου Beowulf.
  Το σημαντικότερο πλεονέκτημά του είναι ότι εγκαθιστά από μόνο του όλο το απαραίτητο λογισμικό για την δημιουργία του cluster, γλιτώνοντας έτσι τον διαχειριστή του cluster από αρκετή δουλεια.

5.6 Υπολογιστικά Πλέγματα (Grid Computing)

Το Grid Computing είναι μια μορφή κατανεμημένης επεξεργασίας που επιτρέπει σε πολλά υπολογιστικά συστήματα να δουλεύουν μαζί για την επίλυση ενός προβλήματος. Τα υπολογιστικά συστήματα που συμμετέχουν στην επίλυση του προβλήματος λέμε ότι σχηματίζουν ένα grid (πλέγμα). Το grid λογίζεται από το εξωτερικό του περιβάλλον, περισσότερο ως ένα δίκτυο συνδεδεμένων υπολογιστικών συστημάτων που δουλεύουν για την επίλυση ενός προβλήματος, παρά ως ένα συνολικό ενιαίο υπολογιστικό σύστημα.

Επειδή συχνά δημιουργείται σύγχυση ανάμεσα στα clusters και στα grids, παρατίθενται στη συνέχεια οι σημαντικότερες διαφορές τους:

- Οι κόμβοι του cluster επικοινωνούν πολύ συχνά μεταξύ τους και ανταλλάσουν δεδομένα για την επίλυση του προβλήματος. Αντίθετα η επικοινωνία των κόμβων του grid περιορίζεται στην ανάθεση του προβλήματος και στην επιστροφή των αποτελεσμάτων.
- Οι κόμβοι του cluster επικοινωνούν πάνω από ένα αφοσιωμένο σε αυτούς δίκτυο διασύνδεσης, σε αντίθεση με τους κόμβους του grid που χρησιμοποιούν οποιοδήποτε δίκτυο διασύνδεσης, ακόμα και το internet.
- Στα clusters οι κόμβοι έχουν μεγάλη ομοιογένεια καθώς σίγουρα τρέχουν το ίδιο λειτουργικό σύστημα και κατά πάσα πιθανότητα αποτελούνται από το

ίδιο hardware, σε αντίθεση με τους κόμβους του grid που μπορούν να διαθέτουν οποιοδήποτε λειτουργικό σύστημα και hardware, αρκεί να υποστηρίζονται από το λογισμικό του Grid.

- Οι κόμβοι ενός cluster συνήθως βρίσκονται όλοι μαζί σε ένα χώρο, ενώ οι κόμβοι του grid μπορούν να βρίσκονται οπουδήποτε.
- Στην περίπτωση των cluster οι κόμβοι του εκτελούν συνήθως μόνο τις εργασίες που έχει αναλάβει το cluster ως σύνολο, ενώ στα grid οι κόμβοι μπορούν εκτός από το πρόβλημα στο οποίο δουλεύουν ως μέρος του grid, μπορούν να επιτελέσουν και οποιαδήποτε άλλη εργασία.
- Το cluster συμπεριφέρεται ως ένα συνολικό σύστημα και οι πόροι του συστήματος αυτού διαχειρίζονται με μία κεντρικοποιημένη προσέγγιση. Στην περίπτωση των grid κάθε υπολογιστικό σύστημα είναι ανεξάρτητο και διαχειρίζεται το ίδιο τους πόρους του.

Έχει ενδιαφέρον να τονιστεί πως ένα cluster μπορεί να συμμετέχει σε ένα grid ως ένα ενιαίο σύστημα και να συμβάλει με την μεγάλη του υπολογιστική ισχύ στην επίλυση του προβλήματος εξ αιτίας του οποίου έχει δημιουργηθεί το Grid.

Οι περιοχές των προβλημάτων στα οποία βρίσκουν εφαρμογή τα grid είναι: Προσομοιώσεις, Υπολογιστική Βιολογία και Ανάλυση Σημάτων και Εικόνων.

Κεφάλαιο 6.
Υποδομή και Εγκατάσταση

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα περιγράψουμε την δομή του συστήματος που χρησιμοποιήθηκε για να γίνουν οι πειραματικές καταγραφές και θα αναφέρουμε την διαδικασία εγκατάστασης όλων των τμημάτων του. Επίσης θα περιγραφεί το virtualization γενικότερα, καθώς το πρόγραμμα virtualization που χρησιμοποιήθηκε, δηλαδή το Xen.

6.2 Δομή Συστήματος

Όπως έχουμε αναφέρει ξανά, ένας από τους σκοπούς μας σε αυτή την διπλωματική είναι να μπορέσουμε να μελετήσουμε πειραματικά μέσω καταγραφών την συμπεριφορά δύο συστημάτων αρχείων κοινού μέσου χωρίς την χρήση του hardware που θα απαιτούσε μια τέτοια μελέτη.

Για να μπορέσουμε να μελετήσουμε τα συστήματα αυτά αρχείων θα πρέπει να έχουμε ένα σύνολο από κόμβους οι οποίοι:

- θα τρέχουν το λογισμικό που θα διαχειρίζεται το εκάστοτε σύστημα αρχείων
- θα επικοινωνούν πάνω από ένα δίκτυο διασύνδεση
- θα έχουν πρόσβαση σε ένα κοινό αποθηκευτικό χώρο

Το hardware που απαιτείται για να γίνει μια τέτοια μελέτη είναι:

- Ένας υπολογιστής για κάθε κόμβο ο οποίος θα χρησιμοποιεί τα συστήματα αρχείων.
- Ένα δίκτυο διασύνδεσης όπως το Ethernet, ή κάποιο υψηλών επιδόσεων δίκτυο διασύνδεση, όπως τα δίκτυα Myrinet και Infiniband. Τα δύο τελευταία χρησιμοποιούνται ως δίκτυα διασύνδεσης σε clusters Υψηλών Υπολογιστικών Επιδόσεων.
  Κάθε κόμβος που θα χρησιμοποιεί το δίκτυο θα πρέπει, να διαθέτει μια ανάλογη κάρτα δικτύου και να συνδέεται σε μια δικτυακή συσκευή όπως ένας μεταγωγέας ή ένας δρομολογητής.
- Για να έχει νόημα η χρησιμοποίηση ενός συστήματος αρχείων κοινού μέσου ή ενός παράλληλου συστήματος αρχείων, θα πρέπει να υπάρχουν αποθηκευτικά μέσα στα οποία έχουν πρόσβαση πολλαπλοί κόμβοι. Τα αποθηκευτικά αυτά μέσα (συνήθως σκληροί δίσκοι) θα πρέπει να ομαδοποιηθούν σε ένα ειδικό δίκτυο αποθηκευτικού χώρου, δηλαδή ένα SAN. Η ομαδοποίηση δίσκων σε SAN απαιτεί ειδικά καλώδια και συσκευές διασύνδεσης. Οι κόμβοι που θα έχουν άμεση πρόσβαση στο SAN θα πρέπει να διαθέτουν μια επιπλέον ειδική κάρτα δικτύου για το SAN και να συνδέονται στις συσκευές διασύνδεσης του SAN. Ορισμένοι κόμβοι που θα χρησιμοποιούν το σύστημα αρχείων, μπορεί να μην έχουν άμεση πρόσβαση στο SAN, αλλά η πρόσβασή τους στον αποθηκευτικό χώρο να γίνεται μέσω των κόμβων που διαθέτουν άμεση

πρόσβαση ακολουθώντας το μοντέλο πρόσβασης NBD. Σημειώνουμε εδώ ότι αυτού του είδους η πρόσβαση θα πρέπει να υποστηρίζεται ρητά από το σύστημα αρχείων προς μελέτη, προκειμένου να είναι εφικτή η χρήση του.

Εφόσον στόχος μας είναι να μην χρησιμοποιήσουμε αυτό το hardware θα πρέπει να κάνουμε χρήση της τεχνολογίας του virtualization προκειμένου να το αναπαραστήσουμε εικονικά.
Στην επόμενη ενότητα περιγράφεται το virtualization καθώς και το λογισμικό Xen που αποτέλεσε το περιβάλλον virtualization που χρησιμοποιήθηκε στην μελέτη αυτή.
Χρησιμοποιώντας το Xen ως περιβάλλον που παρέχει virtualization, πετυχαίνουμε να αναπαραστήσουμε το hardware ως εξής:

- Μπορούμε να δημιουργήσουμε πλήθος εικονικών κόμβων που θα τρέχουν το λειτουργικό σύστημα της αρεσκείας μας χωρίς κόπο. Οι κόμβοι αυτοί τρέχουν ως εικονικές μηχανές (domU στην ορολογία του Xen) πάνω από το πραγματικό μηχάνημα (dom0 στην ορολογία του Xen).
- Κάνοντας χρήση μιας ειδικής δυνατότητας που παρέχει το Xen, κάθε εικονικός κόμβος μπορεί να πάρει μια IP διεύθυνση του δικτύου στο οποίο είναι συνδεδεμένο το πραγματικό μηχάνημα. Επιπλέον η MAC διεύθυνση της εικονικής κάρτας δικτύου, μπορεί να καθοριστεί ελεύθερα από τον χρήστη. Το αποτέλεσμα της παραπάνω δυνατότητας είναι ότι καθίσταται εύκολη η δικτυακή επικοινωνία μεταξύ των εικονικών κόμβων.
- Στο Xen κάθε κόμβος μπορεί να χρησιμοποιήσει ως εικονικό του σκληρό δίσκο είτε ένα αρχείο, είτε ένα partition του πραγματικού σκληρού δίσκου. Με μια πολύ απλή ρύθμιση, το Xen επιτρέπει σε πολλούς εικονικούς κόμβους να μοιράζονται έναν εικονικό σκληρό δίσκο. Με αυτό τον απλό τρόπο μπορούμε να δημιουργήσουμε ένα εικονικό SAN. Σαφώς ισχύει ότι θα πρέπει εκτός από τον κοινό εικονικό δίσκο που θα προσομοιώνει το SAN, κάθε κόμβος να διαθέτει έναν εικονικό σκληρό δίσκο για αποκλειστική δική του χρήση όπου θα εγκατασταθεί το λειτουργικό του σύστημα.

6.3 Virtualization

Virtualization είναι η διαδικασία ή μεθοδολογία για την διαίρεση των πόρων ενός υπολογιστικού συστήματος σε πολλά περιβάλλοντα εκτέλεσης. Η διαίρεση αυτή είναι εφικτή με τη χρησιμοποίηση διαφόρων τεχνολογιών και τεχνικών όπως, ειδική υποστήριξη μέσω hardware, μερική ή ολική προσομοίωση συστημάτων, emulation (δηλαδή την διαδικασία στην οποία ένα σύστημα A παρέχει τις λειτουργίες ενός συστήματος B με αποτέλεσμα το σύστημα A να συμπεριφέρεται όπως το B), πολυπλεξία χρόνου κ.α.

Υπάρχουν τρία είδη virtualization.

- Platform Virtualization, όπου το λειτουργικό σύστημα διαχωρίζεται από το υλικό του υπολογιστικού συστήματος
- Resource Virtualization, όπου προσομοιώνεται η λειτουργία ορισμένων πόρων του υπολογιστικού συστήματος όπως αποθηκευτικά μέσα
- Application Virtualization, που αφορά την δυνατότητα εκτέλεσης μεμονομένων εφαρμογών σε πολλαπλά υπολογιστικά περιβάλλοντα.

Εδώ θα ασχοληθούμε μόνο με το Platform Virtualization. Το Platform Virtualization μέσω της απομόνωσης του λειτουργικού συστήματος από το υλικό επιτρέπει πρώτον να εκτελούνται πολλά λειτουργικά συστήματα ταυτόχρονα σε έναν υπολογιστή και δεύτερον να εκτελείται το λειτουργικό σε περιβάλλοντα που κανονικά δεν θα μπορούσε να τρέξει.

Δύο είναι κύριες μέθοδοι με την οποίες παρέχεται το Platform Virtualization. Το Full Virtualization και το Paravirtualization.

Στο Full Virtualization το λογισμικό που αναλαμβάνει το virtualization κάνει μια πλήρη αφαίρεση των πόρων του υπολογιστικού συστήματος δημιουργώντας ένα πλήρως εικονικό και ελεγχόμενο περιβάλλον μέσα στο οποίο το μπορούν να τρέξουν "guest" λειτουργικά συστήματα. Με αυτή την προσέγγιση δεν είναι απαραίτητη ουδεμία μετατροπή στα λειτουργικά συστήματα που τρέχουν στους εικονικούς κόμβους. Σε αυτή την περίπτωση το λειτουργικό σύστημα δεν γνωρίζει ότι εκτελείται σε ένα εικονικό περιβάλλον και ως εκ τούτου συμπεριφέρονται σαν να έτρεχαν σε ένα κανονικό περιβάλλον.
Το λογισμικό που δημιουργεί το full virtualization περιβάλλον μπορεί να εκμεταλλευτεί ειδικές τεχνολογίες που παρέχουν οι επεξεργαστές (όπως οι τεχνολογίες
IVT και Pacifica από την Intel και την AMD αντίστοιχα) προκειμένου να γίνει πιο εύκολη (και σε μερικές περιπτώσεις πιο γρήγορη) η εργασία του. Η υποστήριξη όμως αυτή δεν είναι συνήθως απαραίτητη από το λογισμικό που κάνει full virtualization.
Τα πιο γνωστά προγράμματα που κάνουν full virtualization είναι μεταξύ των άλλων το Sun VirtualBox, Microsoft Virtual Server, το Virtual Iron και η σειρά προϊόντων της εταιρείας VMware.

Στο Paravirtualization παρέχεται στο λειτουργικό σύστημα που εκτελείται στον εικονικό κόμβο, ένα ειδικό προγραμματιστικό interface που αφορά την πρόσβαση στους πόρους του συστήματος. Η μέθοδος αυτή έρχεται να συμπληρώσει ή/και να αντικαταστήσει την χρήση εικονικών πόρων. Συνεπώς στο Paravirtualization το λειτουργικό σύστημα που εκτελείται σε έναν εικονικό κόμβο πρέπει να υποστεί τροποποιήσεις ώστε να αντιλαμβάνεται το εικονικό περιβάλλον μέσα στο οποίο εκτελείται. Οι τροποποιήσεις αυτές που αναφέραμε έχουν και τον επιπλέον στόχο να καθιστούν σπάνια την εκτέλεση assembly εντολών που δεν εκτελούνται γρήγορα σε virtual περιβάλλον.
Το γνωστότερο προϊόν που χρησιμοποιεί αυτή την τεχνολογία είναι το Xen.

Τα πλεονεκτήματα που συνεπάγεται η χρήση του virtualization είναι τα εξής:

- Επιτρέπεται η μείωση του αριθμού των πραγματικών servers που είναι τρέχουν ορισμένες εφαρμογές όταν ορισμένοι εξ αυτών διεκπεραιώνουν μικρό μέρος της συνολικής εργασίας. Αυτό είναι εφικτό καθώς οι servers που δεν αναλαμβάνουν μεγάλο μέρος της εργασίας μπορούν να τρέχουν ως εικονικές μηχανές. Τα πλεονεκτήματα που επιφέρει η συγκεκριμένη προσέγγιση είναι η μείωση του απαραίτητου εξοπλισμού και η πιο εύκολη διαχείριση των servers.
- Μπορούν να δημιουργηθούν εικονικά συστήματα που θα τρέχουν εφαρμογές που είτε δεν τρέχουν σε υπολογιστές με νεότερο hardware ή/και λειτουργικό σύστημα είτε η εκτέλεσή τους σε νεότερους υπολογιστές θα αποτελούσε σπατάλη των πόρων τους.
- Μπορούν να εκτελεστούν άφοβα εφαρμογές να μην είναι πλήρως έμπιστες ή είναι γνωστό ότι δημιουργούν προβλήματα. Η φύση των εφαρμογών αυτών ενδέχεται να καταστήσει την εκτέλεσή τους σε πραγματικό υπολογιστικό

σύστημα μια επισφαλής επιλογή.

- Μειώνονται οι υπολογιστές που είναι απαραίτητοι σε περιβάλλοντα ανάπτυξης εφαρμογών που έχουν σκοπό να τρέχουν σε πολλαπλά λειτουργικά συστήματα. Ο έλεγχος σωστής λειτουργίας των υπό ανάπτυξη εφαρμογών μπορεί να γίνει στα εικονικά μηχανήματα που μπορούν να τρέχουν οποιοδήποτε λειτουργικό χρειάζεται.

6.3.1 Xen

Το Xen αποτελεί ένα σύστημα λογισμικού που επιτρέπει σε πολλά εικονικά μηχανήματα να τρέχουν σε έναν υπολογιστή. Σε αντίθεση με άλλα συστήματα λογισμικού που παρέχουν την ίδια λειτουργία, το Xen δεν αποτελεί μια εφαρμογή που τρέχει πάνω από το λειτουργικό σύστημα, αλλά έναν τροποποιημένο πυρήνα ενός εκ των λειτουργικών Linux, NetBSD και Solaris. Οι τροποποιήσεις στον πυρήνα επιτρέπουν να γίνεται διαχείριση των εικονικών μηχανημάτων.

Ο τροποποιημένος αυτός πυρήνας τρέχει στο πιο προνομιακό επίπεδο που παρέχει ο επεξεργαστής και συνεπώς έχει άμεση πρόσβαση στο υλικού του υπολογιστή. Τα εικονικά μηχανήματα τρέχουν στο αμέσως επόμενο προνομιακά, επίπεδο του επεξεργαστή.

Στην ορολογία του Xen, το λειτουργικό σύστημα που τρέχει στο πιο προνομιακό επίπεδο και διαχειρίζεται τα εικονικά μηχανήματα ονομάζεται dom0 ενώ τα εικονικά μηχανήματα ονομάζονται domU.

Υπάρχουν δύο τρόποι δημιουργίας του εικονικού περιβάλλοντος για τα εικονικά μηχανήματα.

Ο πρώτος είναι να χρησιμοποιηθεί paravirtualized περιβάλλον και συνεπώς για να μπορεί να δημιουργηθεί ένα domU, θα πρέπει το λειτουργικό σύστημα του να είναι τροποποιημένο ώστε να γνωρίζει ότι τρέχει σε εικονικό περιβάλλον. Τα λειτουργικά που έχουν υποστεί τις απαραίτητες τροποποιήσεις και μπορούν να λειτουργήσουν στο paravirtualized περιβάλλον του xen (στην έκδοση 3), είναι τα: Linux, NetBSD, FreeBSD και Solaris.

Ο δεύτερος τρόπος είναι να χρησιμοποιηθεί το Xen σε συστήματα που διαθέτουν επεξεργαστή με τεχνολογία Intel VT ή AMD Pacifica ώστε να δημιουργηθεί ένα fully virtualized περιβάλλον για τα εικονικά μηχανήματα. Με αυτό τον τρόπο είναι εφικτή η δημιουργία εικονικών μηχανημάτων που τρέχουν (σχεδόν) οποιοδήποτε λειτουργικό σύστημα χωρίς την ανάγκη τροποποίησής του.

Τα κύρια πλεονεκτήματα που παρέχει το Xen ως μια πλατφόρμα virtualization είναι τα παρακάτω:

- Σε paravirtualized περιβάλλον οι εικονικές μηχανές εκτελούνται με επιδόσεις που είναι κάτω από 5% μικρότερες από ότι σε πραγματικό περιβάλλον. Αυτές οι επιδόσεις είναι κατά πολύ ανώτερες άλλων συστημάτων που δεν χρησιμοποιούν virtualization αφού σε αυτά οι εικονικές μηχανές μπορεί να εκτελούνται με μείωση επιδόσεων ακόμα και 35%.
- Υποστήριξη πληθώρας λειτουργικών συστημάτων.
- Ευκολία χρήσης.
- Ύπαρξη μεγάλης και δραστήριας κοινότητας.

6.4 Πλεονεκτήματα και Μειονεκτήματα της Χρήσης Εικονικού Hardware

Η χρήση του Xen για να την προσομοίωση του απαραίτητου hardware σε έναν πραγματικό κόμβο έχει συγκεκριμένα πλεονεκτήματα και μειονεκτήματα.
Αρχικά θα εξετάσουμε τα πλεονεκτήματα.

- Το πρώτο και σημαντικότερο πλεονέκτημα είναι η εξάλειψη του κόστους που συνεπάγεται η χρήση πολλών κόμβων και της χρήσης του SAN. Το κόστος του πραγματικού hardware και ιδιαίτερα του SAN που απαιτείται για μια τέτοια μελέτη είναι αρκετά υψηλό.
  Με την προσομοίωση του hardware δεν υπάρχει κανένα κόστος επιπλέον κόστος πέραν του κόστους του πραγματικού μηχανήματος.
- Ένα δεύτερο σημαντικό πλεονέκτημα είναι ο μικρότερος χρόνος εγκατάστασης του λογισμικού που απαιτείται για πολλούς κόμβους. Αυτό είναι εφικτό καθώς αφού στηθεί το πρώτο εικονικό μηχάνημα και εγκατασταθεί σε αυτό το απαραίτητο λογισμικό, είναι δυνατόν να αντιγραφεί το εικονικό μηχάνημα σε ένα άλλο στο οποίο απλά θα πρέπει να αλλαχτούν οι ρυθμίσεις που αφορούν το όνομα και την δικτύωση ώστε να εμφανίζεται στο δίκτυο.
- Ένα τρίτο πλεονέκτημα είναι οι γρήγορες και εύκολες αλλαγές στο configuration των εικονικών μηχανημάτων. Για παράδειγμα αν παραστεί η ανάγκη να προστεθεί σε κάθε εικονικό μηχάνημα ένας σκληρός δίσκος, αρκεί να δημιουργηθεί ένα αρχείο που θα παίζει τον ρόλο του σκληρού δίσκου στα για τα εικονικά μηχανήματα και να προστεθεί μια καταχώριση στα αρχεία που περιγράφουν τα εικονικά μηχανήματα για να δηλωθεί η ύπαρξη του νέου σκληρού δίσκου.

Τα μειονεκτήματα που συνεπάγεται η χρήση της τεχνολογίας virtualization για την προσομοίωση του απαραίτητου hardware σε έναν πραγματικό κόμβο είναι τα εξής:

- Δεν είναι δυνατόν να προσομοιωθεί μεγάλος αριθμός κόμβων από ένα μόνο πραγματικό μηχάνημα. Αυτό διότι κάθε κόμβος αποτελεί ένα εικονικό μηχάνημα που δεσμεύει σημαντική ποσότητα της κεντρικής μνήμης του πραγματικού υπολογιστή. Συνεπώς ο αριθμός των εικονικών μηχανημάτων που μπορούν να προσομοιωθούν σε έναν υπολογιστή εξαρτάται απόλυτα από την ποσότητα μνήμης που απαιτεί κάθε εικονικό μηχάνημα για να τρέξει το απαραίτητο λογισμικό και την συνολική ποσότητα της κεντρικής μνήμης.
- Δεν είναι δυνατόν να υπάρξει μελέτη των χρονικών επιδόσεων των δύο συστημάτων αρχείων καθώς αφενός υπάρχει διαμοιρασμός των επεξεργαστών σε πολλά εικονικά μηχανήματα και αφετέρου τα εικονικά μηχανήματα δεν διαθέτουν πραγματικά αποθηκευτικά μέσα.
  Ως αποτέλεσμα αυτού του μειονεκτήματος, δεν μπορούμε να δούμε στην πράξη τι πλεονέκτημα επιδόσεων δίνει στο GPFS η τεχνική του striping, ή την διαφορά επιδόσεων κατά την χρήση του Data Shipping Mode σε αντίθεση με την απλή πρόσβαση στα δεδομένα.
  Ο περιορισμός αυτός δεν αποτελεί πρόβλημα στην περίπτωση της μελέτης μας, καθώς η φύση της μελέτης αυτής δεν περιλαμβάνει μέτρηση χρονικών επιδόσεων.

6.5 Διαδικασία Εγκατάστασης

Στην παράγραφο αυτή, θα εξηγήσουμε αναλυτικά την διαδικασία εγκατάστασης που ακολουθήθηκε δίνοντας έμφαση στα πιο σημαντικά σημεία και παρέχοντας τις λύσεις που υιοθετήθηκαν για τα πιο σημαντικά προβλήματα που ανέκυψαν.

6.5.1 Πορεία μέχρι την τελική εγκατάσταση

Η διαδικασία της εγκατάστασης διήρκεσε μεγάλο χρονικό διάστημα, είχε πολλές δυσκολίες αλλά έδωσε την δυνατότητα στον γράφοντα να μάθει πολλά πράγματα για τα συστήματα αρχείων GFS και GPFS καθώς και για το λειτουργικό σύστημα Linux. Στην υποπαράγραφο αυτή περιγράφουμε την γενική πορεία που ακολουθήθηκε μέχρι να γίνει η τελική εγκατάσταση.

Κατά την διαδικασίας της εγκατάστασης αλλά και της διεξαγωγής της μελέτης, το λειτουργικό σύστημα που έτρεχε στο πραγματικό μηχάνημα ήταν το Ubuntu 7.04.
Το πραγματικό μηχάνημα αποτέλεσε ο φορητός υπολογιστής του γράφοντος.
Η έκδοση αυτή επιλέχθηκε καθώς είναι αρκετά σταθερή (σε αντίθεση με νεότερες) και συνεργάζεται καλά με περιβάλλοντα virtualization.

Αρχικά θεωρήθηκε σκόπιμο να χρησιμοποιηθεί ως περιβάλλον virtualization το πρόγραμμα VirtualBox που παρέχει η εταιρεία Sun Microsystems (αφού αγόρασε την αρχική κατασκευάστρια Innotek). Η επιλογή αυτή έγινε καθώς το VirtualBox τρέχει ως εφαρμογή του χώρου χρήστη και δεν απαιτεί καμία τροποποίηση στο υφιστάμενο λειτουργικό σύστημα και επιπλέον η χρήση του είναι πολύ απλή.
Με την χρήση του VirtualBox δημιουργήθηκαν τρία εικονικά μηχανήματα που έτρεχαν Ubuntu 7.10 ως λειτουργικό σύστημα. Ο λόγος που επιλέχθηκε το συγκεκριμένο λειτουργικό ήταν ότι είχε στα repositories του το πακέτο RedHat Cluster Suite που περιέχει σε binary μορφή ότι χρειάζεται το GFS για να λειτουργήσει σωστά.
Στο RedHat Cluster Suite, όλοι οι κόμβοι που θα χρησιμοποιήσουν ένα GFS σύστημα αρχείων, θα πρέπει να συμμετέχουν σε ένα RedHat cluster. Η δημιουργία αυτού του cluster γίνεται με την κλήση των ειδικών προγραμμάτων που παρέχονται στο πακέτο αυτό, αφού όμως κάθε κόμβος λάβει ένα αντίγραφο του αρχείου /etc/cluster/cluster.conf.
Λεπτομέρειες για την μορφή του αρχείου cluster.conf θα παρατεθούν αργότερα όταν θα περιγραφεί η εγκατάσταση του GFS σε εικονικό μηχάνημα με χρήση του Xen.
Στο σημείο αυτό μας ενδιέφερε να δημιουργήσουμε ένα GFS σύστημα αρχείων με χρήση όμως του GNBD. Το GNBD ότι είναι η υλοποίηση της RedHat για το NBD μοντέλο πρόσβασης σε δίσκους.

Η παραπάνω περιγραφή συνεπάγεται ότι μόνο ένας από τους τρεις εικονικούς κόμβους του RedHat cluster είχε άμεση πρόσβαση στο εικονικό SAN, ενώ οι άλλοι δύο είχαν πρόσβαση μέσω του GNBD. Με χρήση των κατάλληλων εντολών του GFS δημιουργήθηκε χωρίς πρόβλημα το σύστημα αρχείων. Λεπτομέρειες για την δημιουργία του δίνονται αργότερα που θα περιγράψουμε την εγκατάσταση του GFS σε περιβάλλον Xen.

Αφού είχε γίνει μια πρώτη εγκατάσταση του GFS, έπρεπε στη συνέχεια να γίνει εγκατάσταση του GPFS. Για τις ανάγκες του GPFS, θέλαμε εξ αρχής να μπορέσουμε να δημιουργήσουμε το σύστημα αρχείων πάνω από έναν κοινό αποθηκευτικό χώρο και να μην κάνουμε χρήση του NBD μοντέλου πρόσβασης (στο GPFS ονομάζεται NSD Server).
Εφόσον είχε χρησιμοποιηθεί το VirtualBox ως πλατφόρμα virtualization με επιτυχία στην περίπτωση του GFS, ξεκινήσαμε να το χρησιμοποιούμε και για την περίπτωση του GPFS. Αφού πειραματιστήκαμε με αυτό και διαβάσαμε εκτενώς το documentation του, διαπιστώσαμε πως δεν επιτρέπει τον διαμοιρασμό δίσκων σε πολλά μηχανήματα. Δηλαδή το VirtualBox όταν αναθέτει έναν δίσκο σε κάποιο μηχάνημα, δεν επιτρέπει την ανάθεσή του σε άλλο. Για το λόγο αυτό το VirtualBox είναι ακατάλληλο για την δημιουργία εικονικού SAN περιβάλλοντος.
Έπρεπε να επιλεχθεί ένα διαφορετικό περιβάλλον virtualization που θα επέτρεπε να μοιράζεται ένας εικονικός δίσκος σε πολλά εικονικά μηχανήματα.
Το περιβάλλον αυτό βρέθηκε στο Xen. Η εγκατάστασή του αν και περιλαμβάνει την εγκατάσταση ενός ειδικού πυρήνα για το Linux, έγινε χωρίς κανένα πρόβλημα.
Όπως επισημάνθηκε και στην ενότητα 6.3.1, το Xen λειτουργεί και με Paravirtualization και με Full Virtualization. Επειδή το GPFS δεν είναι συμβατό με πυρήνες που είναι τροποποιημένοι ώστε να χρησιμοποιηθούν ως λειτουργικά συστήματα σε domU μηχανήματα, έπρεπε να χρησιμοποιηθεί Full Virtualization. Όπως αναφέρεται και στην ενότητα 6.3.1, για να χρησιμοποιηθεί το Full Virtualization στο Xen, απαιτείται ειδική υποστήριξη από τον επεξεργαστή του πραγματικού μηχανήματος. Στην περίπτωσή μας η υποστήριξη αυτή υπήρχε.
Για την περιγραφή κάθε εικονικού μηχανήματος το Xen χρησιμοποιεί ένα αρχείο που πρέπει να τοποθετείται κάτω από τον κατάλογο /etc/xen. Ένα παράδειγμα ενός τέτοιου αρχείου είναι το παρακάτω:

```
kernel = "/usr/lib/xen/hvmloader"
builder = "hvm"
memory = 512
name = "debian1"
vcpus = 1
vif = [mac=aa:00:00:00:00:01, bridge=xen-br0']
disk = [ 'file:/xen/debian1.disk,ioemu:hda,w',
'file:/xen/shared.disk,ioemu:hdb,w!',
'phy:/dev/cdrom,ioemu:hdc:cdrom,r' ]
vnc=1
```

Οι δύο πρώτες επιλογές αφορούν τον τρόπο με τον οποίο θα φορτωθεί το εικονικό μηχάνημα, η τρίτη είναι η μνήμη του πραγματικού συστήματος που θα δεσμεύεται από το εικονικό μηχάνημα, η τέταρτη είναι το όνομα που θα έχει το εικονικό μηχάνημα (μέσω αυτού του ονόματος το ξεκινάμε και το σταματάμε), η πέμπτη είναι ο αριθμός των επεξεργαστών που θα διαθέτει, η έκτη αφορά την δικτύωση, η έβδομη είναι οι εικονικοί δίσκοι και η τελευταία δείχνει τον δίσκο από τον οποίο θα κάνει boot το εικονικό μηχάνημα.
Δύο στοιχεία του αρχείου πρέπει να προσεχθούν ιδιαίτερα.
Πρώτον την ρύθμιση που αφορά την δικτύωση. Βλέπουμε ότι χρησιμοποιείται η ρύθμιση που δίνει την δυνατότητα στο εικονικό μηχάνημα να πάρει μια πραγματική διεύθυνση του δικτύου στο οποίο είναι συνδεδεμένο το πραγματικό μηχάνημα.

Επιπλέον καθορίζεται από τον δημιουργό του αρχείου η MAC διεύθυνση που θα έχει η εικονική κάρτα δικτύου του εικονικού μηχανήματος.
Δεύτερο και πιο σημαντικό στοιχείο είναι η χρήση των σκληρών δίσκων. Και οι δύο εικονικοί σκληροί δίσκοι που χρησιμοποιεί το εικονικό μηχάνημα είναι στην ουσία αρχεία του πραγματικού συστήματος.
Ο πρώτος σκληρός δίσκος είναι ο κύριος και χρησιμοποιείται για την εγκατάσταση του λειτουργικού όπως δείχνει και η γραμμή root. Ο δεύτερος σκληρός δίσκος παρουσιάζει ενδιαφέρον εξ' αιτίας του w! που περιέχει. Το w! σημαίνει το αρχείο που αναπαριστά τον σκληρό δίσκο μπορεί να χρησιμοποιηθεί από πολλά εικονικά μηχανήματα. Αυτή την ρύθμιση χρησιμοποιούμε για να έχουν τα εικονικά μηχανήματα πρόσβαση σε ένα εικονικό SAN.
Η χρήση του w! από το Xen είναι μια συνετή επιλογή, καθώς με αυτό τον τρόπο ο χρήστης πρέπει να δηλώνει ρητά ότι θέλει να επιτρέπεται η κοινή πρόσβαση σε εικονικούς δίσκους από τα εικονικά μηχανήματα (η κανονική επιλογή χωρίς διαμοιρασμό αντικαθιστά το w με το w!) αποφεύγοντας έτσι κατά λάθος διαμοιρασμούς δίσκων που θα σήμαινε και καταστροφή του τοπικού συστήματος αρχείων που ενδεχομένως περιέχουν.

Με χρήση λοιπόν του Full Virtualization δημιουργήθηκαν τέσσερα μηχανήματα με λειτουργικό σύστημα Debian στην έκδοση 4.
Για να είναι δυνατή η εγκατάσταση του GPFS σε ένα υπολογιστή που τρέχει λειτουργικό σύστημα Linux, απαιτείται η μεταγλώττιση του πηγαίου κώδικά του. Το GPFS αποτελεί λογισμικό που τρέχει στο kernel space και χρησιμοποιεί πολλά κομμάτια του πυρήνα. Επειδή τα κομμάτια του πυρήνα αλλάζουν συχνά από την μία έκδοση στην άλλη, η εύρεση κατάλληλου πυρήνα για να γίνει η μεταγλώττιση του GPFS, είναι δύσκολη. Το γεγονός αυτό δεν ισχύει μόνο για το GPFS, αλλά για όλα τα κομμάτια κώδικα που προτίθενται να λειτουργήσουν στο kernel space και απαιτείται η μεταγλώττισή τους. Στα κομμάτια αυτά λογισμικού εντάσσονται και τα συστήματα αρχείων.
Στην περίπτωσή της μελέτης μας υπήρχε ένας επιπλέον περιορισμός που αφορούσε τον πυρήνα. Έπρεπε ο πυρήνας ο οποίος θα χρησιμοποιούνταν για την μεταγλώττιση του GPFS (και του GFS), να υποστηρίζει το blktrace. Το blktrace είναι ένα από τα προγράμματα που χρησιμοποιήθηκαν στην μελέτη μας για να γίνει tracing των I/O διαδικασιών. Περιγράφεται στην ενότητα 7.4.
Τελικά βρέθηκε ότι ο κατάλληλος πυρήνας ήταν ο 2.6.16 (στον οποίο προστέθηκε ένα patch για την υποστήριξη του blktrace)
Η μεταγλώττιση του GPFS σε αυτόν τον πυρήνα έγινε χωρίς κανένα απολύτως πρόβλημα.
Αφού πλέον είχαμε ένα σταθερό σύστημα στο οποίο μπορούσε να εγκατασταθεί το GPFS, δημιουργήσαμε τρεις νέους κόμβους, αντιγράφοντας τρεις φορές το αρχείο που έπαιζε το ρόλο του σκληρού δίσκου του πρώτου κόμβου δημιουργώντας έτσι έναν εικονικό σκληρό δίσκο για κάθε έναν από τους τρεις νέους κόμβους. Ο εικονικός σκληρός δίσκος κάθε κόμβου περιείχε όλο το απαραίτητο λογισμικό για το GPFS καθώς αυτό είχε εγκατασταθεί στον πρώτο κόμβο από τον οποίο έγινε η αντιγραφή. Οι μόνες ρυθμίσεις που έπρεπε να γίνουν σε κάθε κόμβο αφορούσαν το όνομά και την δικτύωσή του.
Η διαδικασία της εγκατάστασης συνεχίστηκε προκειμένου να εγκατασταθεί το GFS στο ίδιο σύστημα στο οποίο εγκαταστάθηκε το GPFS.

Μετά από πολλές απόπειρες, κατέστη δυνατή η μεταγλώττιση της έκδοσης 1.02 του RedHat Cluster Suite αφού του προστέθηκαν δύο patch που του επέτρεπαν να λειτουργεί με πυρήνα 2.6.16.
Η ίδια διαδικασία επαναλήφθηκε και για τους άλλους τρεις κόμβους.

6.5.2 Τελική Εγκατάσταση και δημιουργία των Συστημάτων Αρχείων

Το συστήματα στα οποία έγινε η εγκατάσταση των δύο συστημάτων αρχείων, ήταν όπως είπαμε τέσσερις κόμβοι που διέθεταν λειτουργικό σύστημα Debian στην έκδοση 4 αλλά έτρεχαν πυρήνα 2.6.16 του kernel.org που διέθετε patch για την υποστήριξη του block tracing. Οι κόμβοι αυτοί έτρεχαν ως εικονικές μηχανές σε Xen περιβάλλον. Η έκδοση του Xen ήταν η έκδοση 3.02 και χρησιμοποιούσε τροποποιημένα πυρήνα Ubuntu 7.04 ως πυρήνα του Hypervisor.
Θα περιγράψουμε αμέσως παρακάτω το στήσιμο των δύο συστημάτων αρχείων.

GPFS

Αφού γίνει η μεταγλώττιση του GPFS σε κάθε ένα κόμβο, ακολουθεί η διαδικασία δημιουργίας του GPFS cluster μεταξύ των κόμβων και του συστήματος αρχείων.
Υπάρχουν ορισμένα προκαταρκτικά που θα πρέπει να γίνουν προκειμένου να είναι δυνατή η δημιουργία του GPFS cluster. Θα πρέπει στο αρχείο /etc/hosts κάθε κόμβου να υπάρχουν τα ονόματα και οι IP διευθύνσεις όλων των κόμβων που θα συμμετέχουν στο cluster. Αυτό είναι απαραίτητο γιατί πρέπει όλοι οι κόμβοι να γνωρίζουν την ύπαρξη όλων των άλλων κόμβων. Στη συνέχεια θα πρέπει κάθε κόμβος του cluster να μπορεί να κάνει login και αντιγραφή αρχείων με χρήση των ssh και rcp αντίστοιχα, χωρίς την ανάγκη χρήσης password. Αυτό είναι απαραίτητο καθώς όταν τρέχει το GPFS, ορισμένες διαχειριστικές διαδικασίες απαιτούν την ανταλλαγή αρχείων μεταξύ των κόμβων. Για να είναι δυνατή χρήση των ssh και rcp μεταξύ όλων των κόμβων χωρίς password θα πρέπει το δημόσιο κλειδί κάθε κόμβου να βρίσκεται στο αρχείο ~/.ssh/authorized_keys κάθε άλλου κόμβου.
Αφού γίνουν αυτά τα προκαταρκτικά ακολουθεί η δημιουργία του cluster.
Η δημιουργία του cluster γίνεται με την εντολή mmcrcluster και απαιτεί ορισμένες πληροφορίες από τον χρήστη. Οι πληροφορίες αυτές είναι:

- Το όνομα του cluster. Θα πρέπει να επιλεχθεί ένα όνομα από τον χρήστη προκειμένου να αναγνωρίζεται μοναδικά το cluster. Η πιο σημαντική χρήση του ονόματος είναι για την αποφυγή εκκίνησης δύο στιγμιότυπων του ίδιου cluster, μια κατάσταση που αναφέρεται ως split brain condition.
- Όπως αναφέραμε και στο κεφάλαιο 4 στην περιγραφή του GPFS, κάθε GPFS cluster διαθέτει έναν Primary και έναν Secondary Cluster Manager. Αυτοί οι κόμβοι (ένας κάθε φορά) επιτελούν όλες τις απαραίτητες ενέργειες για την διαχείριση του cluster. Κατά την δημιουργία του cluster ο χρήστης θα πρέπει δηλώσει τα ονόματα των κόμβων που θα αναλάβουν τους ειδικούς αυτούς ρόλους.
- Τα ονόματα των κόμβων που θα συμμετέχουν στο cluster καθώς και τους ειδικούς ρόλους που πιθανόν να μπορούν να αναλάβουν αυτοί οι κόμβοι. Οι ειδικοί αυτοί ρόλοι είναι οι token managers και file system managers. Όλες αυτές οι πληροφορίες αποθηκεύονται σε ένα αρχείο και η διαδρομή του αρχείου αυτού δηλώνεται κατά την κλήση της mmcrcluster. Πρέπει επίσης να αναφέρουμε πως θα πρέπει στο αρχείο αυτό να δηλώνεται

ποιοι κόμβοι είναι απαραίτητοι για την εκκίνηση του cluster. Χωρίς να είναι ενεργοί τουλάχιστον οι μισοί συν ένας από αυτούς τους κόμβους δεν είναι δυνατόν να ξεκινήσει το cluster, άσχετα με το πόσοι άλλοι μη απαραίτητοι κόμβοι είναι ενεργοί. Το γεγονός αυτό βοηθάει να αποφεύγεται το split brain condition που αναφέρθηκε πιο πάνω.

- Υπάρχει πλήθος άλλων πληροφοριών που αφορούν το cluster και μπορούν είτε να καθοριστούν σε αυτό το βήμα, ή σε οποιαδήποτε άλλη στιγμή με χρήση των εντολών mmchconfig και mmchcluster.

Μετά την δημιουργία του cluster, σειρά παίρνει η δήλωση του αποθηκευτικού χώρου που θα χρησιμοποιηθεί για την δημιουργία GPFS συστημάτων αρχείων. Όπως έχουμε ξαναπεί, το GPFS για την δημιουργία συστημάτων αρχείων μπορεί να χρησιμοποιήσει δίσκους ενός SAN στο οποίο όλοι οι κόμβοι του cluster έχουν κοινή πρόσβαση ή τοπικούς δίσκους οποιουδήποτε κόμβου χρησιμοποιώντας το μοντέλο πρόσβασης NBD για πρόσβαση από τους άλλους κόμβους.
Με την χρήση της εντολή mmcrnsd που δέχεται ως παράμετρο ένα αρχείο που περιγράφει όλους τους προς χρησιμοποίηση από το GPFS δίσκους, γίνεται η δήλωση του αποθηκευτικού χώρου. Στο αρχείο αυτό δηλώνεται αν ο δίσκος ανήκει σε SAN ή είναι τοπικά συνδεδεμένος σε κάποιο κόμβο, το αρχείο συσκευής του καταλόγου /dev/ του Linux που τον περιγράφει και του δίνεται ένα μοναδικό όνομα για χρήση από το GPFS.
Επίσης μπορεί να δηλωθεί σε ποια ομάδα δίσκων θα τοποθετηθεί ο δίσκος.
Όπως έχουμε πει και στην ενότητα 4.8, το GPFS επιτρέπει οι δίσκοι να ομαδοποιούνται με κριτήρια που επιλέγει ο διαχειριστής του cluster και πάνω σε αυτούς να εφαρμόζονται διάφορες πολιτικές.

Τέλος θα πρέπει να δημιουργηθούν τα συστήματα αρχείων. Η δημιουργία κάθε συστήματος αρχείων γίνεται μέσω της εντολής mmcrfs. Οι πληροφορίες που είναι απαραίτητες για την δημιουργία ενός συστήματος αρχείων και παίρνουν την μορφή παραμέτρων προς την εντολή mmcrfs είναι:

- Οι δίσκοι πάνω στους οποίους θα στηθεί το σύστημα αρχείων. Οι δίσκοι αυτοί θα πρέπει να έχουν δηλωθεί στο προηγούμενο βήμα. Όπως συμβαίνει για όλα τα συστήματα αρχείων, σε κάθε partition μπορεί να υπάρχει μόνο ένα σύστημα αρχείων.
- Το όνομα του συστήματος αρχείων. Θα πρέπει να δοθεί ένα μοναδικό όνομα στο σύστημα αρχείων προκειμένου να είναι δυνατή η διαχείρισή του.
- Ο μέγιστος αριθμός κόμβων του cluster που θα χρησιμοποιούν ταυτόχρονα το σύστημα αρχείων (δηλαδή θα το έχουν κάνει mount). Η παράμετρος αυτή είναι απαραίτητη προκειμένου να δημιουργηθεί ο αντίστοιχος αριθμός logs. Κάθε log θα χρησιμοποιηθεί από έναν κόμβο που θα χρησιμοποιήσει το σύστημα αρχείων με τον τρόπο που έχει περιγραφεί σε προηγούμενη παράγραφο.
- Η θέση μέσα στον κατάλογος αρχείων στην οποίο θα γίνεται mount το σύστημα αρχείων σε κάθε κόμβο. Η πληροφορία αυτή χρησιμοποιείται ώστε να γραφτεί στο αρχείο /etc/fstab κάθε κόμβου που συμμετέχει στο σύστημα αρχείων. Με αυτό τον τρόπο το GPFS μπορεί να κάνει αυτόματα mount το σύστημα αρχείων στους κόμβους που συμμετέχουν στο cluster όταν αυτό ξεκινάει. Η αυτοματοποιημένη αυτή διαδικασία λαμβάνει χώρα με τον ίδιο τρόπο για κάθε κόμβο άσχετα αν έχει άμεση πρόσβαση στον αποθηκευτικό χώρο ή χρησιμοποιεί το μοντέλο NBD.

- Υπάρχει πλήθος άλλων ρυθμίσεων που αφορούν το σύστημα αρχείων που είτε μπορούν να καθοριστούν στην εντολή mmcrfs είτε να αλλάξουν σε άλλο χρονικό σημείο με την χρήση της mmchfs.

GFS

Για να μπορέσουν οι κόμβοι να χρησιμοποιήσουν ένα GFS σύστημα αρχείων, θα πρέπει αρχικά να συμμετέχουν σε ένα RedHat Cluster. Για να γίνει αυτό θα πρέπει να υπάρχει σε όλους τους κόμβους το αρχείο /etc/cluster/cluster.conf που περιγράφει το cluster και στη συνέχεια να κληθούν τα προγράμματα που το διαχειρίζονται.
Το αρχείο cluster.conf που χρησιμοποιήθηκε φαίνεται παρακάτω:

```
<?xml version="1.0"?>
<cluster name="gc" config_version="1">

<cman expected_votes="4">
</cman>

<clusternodes>
<clusternode name="det1">
        <fence>
                <method name="f1" nodeid="1">
                        <device name="geo_fence"
ipaddr="192.168.2.55"/>
                </method>
        </fence>
</clusternode>

<clusternode name="det2">
        <fence>
                <method name="f2" nodeid="2">
                        <device name="geo_fence"
ipaddr="192.168.2.56"/>
                </method>
        </fence>
</clusternode>

<clusternode name="det3">
        <fence>
                <method name="f3" nodeid="3">
                        <device name="geo_fence"
ipaddr="192.168.2.6"/>
                </method>
        </fence>
</clusternode>

<clusternode name="det2">
        <fence>
                <method name="f4" nodeid="4">
                        <device name="geo_fence"
ipaddr="192.168.2.8"/>
                </method>
        </fence>
</clusternode>

</clusternodes>
```

```
<fencedevices>
        <fencedevice name="geo_fence" agent="fence_manual"/>
</fencedevices>

</cluster>
```

Όπως φαίνεται το αρχείο αυτό είναι σε μορφή XML.
Σε αυτό το XML περιγράφονται οι κόμβοι που συμμετέχουν στο cluster, πόσοι κόμβοι είναι απαραίτητοι για την λειτουργία του καθώς και τον τρόπο με τον οποίο απόβάλλονται από το cluster όταν τεθούν εκτός λειτουργίας.
Κάθε κόμβος που θα συμμετέχει στο cluster περιγράφεται με ένα tag clusternode. Εκεί βρίσκεται αρχικά το όνομα του κόμβου. Το όνομα που εμφανίζεται στο αρχείο πρέπει να είναι ακριβώς το ίδιο με το όνομα του κόμβου όπως αυτό εμφανίζεται στο δίκτυο διασύνδεσης, δηλαδή το όνομα με το οποίο οι άλλοι κόμβοι μπορούν να κάνουν ssh κτλ.
Προαιρετικά μπορεί να υπάρχει και ο αριθμός των ψήφων που διαθέτει ο κόμβος. Αν δεν υπάρχει τέτοια πληροφορία, ο κόμβος έχει μια ψήφο. Για να μπορέσει να ξεκινήσει το cluster, θα πρέπει οι κόμβοι που είναι ενεργοί και θα συμμετέχουν στο cluster να συγκεντρώνουν αριθμό ψήφων ίσο με το συνολικό αριθμό των ψήφων που έχουν όλοι οι κόμβοι που περιγράφονται στο αρχείο συν ένα. Με αυτό τον τρόπο αποφεύγεται η πιθανότητα εκκίνησης δύο στιγμιότυπων του ίδιου cluster, που όπως είπαμε και πιο πριν αναφέρεται ως split brain condition.
Επίσης για κάθε κόμβο θα πρέπει υποχρεωτικά να δηλώνεται η μέθοδος που θα χρησιμοποιηθεί για να αποβληθεί ο κόμβος από το cluster σε περίπτωση που τεθεί εκτός λειτουργίας, δηλαδή η μέθοδος του fencing που θα χρησιμοποιηθεί. Στην περίπτωσή μας χρησιμοποιείται χειροκίνητο fencing.

Όπως και στην περίπτωση του GPFS, πριν γίνει εκκίνηση του cluster, θα πρέπει όλοι οι κόμβοι να γνωρίζουν την ύπαρξη όλων των άλλων κόμβων (μέσω των εγγραφών στο αρχείο /etc/hosts) και να μπορούν να κάνουν σε αυτούς ssh και rcp χωρίς την χρήση password. Μια επιπλέον απαίτηση που έχει το GFS είναι ότι πρέπει όλοι οι κόμβοι του cluster να έχουν συγχρονισμένα τα ρολόγια τους.

Αφού γίνουν οι απαραίτητες προετοιμασίες, ακολουθεί η εκκίνηση του cluster. Σημειώνουμε ότι το GFS δεν έχει διακριτή διαδικασία δημιουργίας του cluster όπως στο GPFS. Αρκεί κάθε κόμβος που θα συμμετέχει να διαθέτει το απαραίτητο λογισμικό, να είναι ρυθμισμένος σωστά όσον αφορά τα προκαταρκτικά και να περιέχει ένα αντίγραφο το cluster.conf.
Για να εκκινήσει το RedHat cluster πρέπει αρχικά κάθε κόμβος να φορτώσει τα απαραίτητα modules και στη συνέχεια να εκκινήσει τα σωστά προγράμματα που παρέχει η εγκατάσταση του GFS.
Αφού ξεκινήσει το cluster, μπορούν να δημιουργηθούν τα συστήματα αρχείων.
Όπως έχουμε αναφέρει και στις διαφορές του GFS και το GPFS, στο πρώτο η δημιουργία συστημάτων αρχείων γίνεται πάνω από σε μόνο δίσκο ή λογικό partition. Η δημιουργία ενός συστήματος αρχείων μπορεί να γίνει από οποιονδήποτε κόμβο έχει άμεση πρόσβαση στο δίσκο ή το λογικό partition.
Η δημιουργία κάθε συστήματος αρχείων γίνεται μέσω της εντολής gfs_mkfs. Οι πληροφορίες που είναι απαραίτητες για την δημιουργία ενός συστήματος αρχείων και παίρνουν την μορφή παραμέτρων προς την εντολή gfs_mkfs είναι:

- Το αρχείο συσκευής που περιγράφει τον δίσκο ή το λογικό partition στο οποίο

θα δημιουργηθεί το σύστημα αρχείων και βρίσκεται στον κατάλογο /dev.

- Το όνομα του συστήματος αρχείων. Θα πρέπει να δοθεί ένα μοναδικό όνομα στο σύστημα αρχείων προκειμένου να είναι δυνατή η διαχείρισή του.
- Ο μέγιστος αριθμός κόμβων του cluster που θα χρησιμοποιούν ταυτόχρονα το σύστημα αρχείων. Η παράμετρος αυτή είναι απαραίτητη προκειμένου να δημιουργηθεί ο αντίστοιχος αριθμός logs.
- Το πρωτόκολλο που θα χρησιμοποιηθεί για τον χειρισμό των κλειδωμάτων που θα διατηρήσουν την συνέπεια του συστήματος αρχείων. Όπως έχουμε αναφέρει το GFS λόγω της modular σχεδίασής του επιτρέπει την χρησιμοποίηση πολλών διαφορετικών πρωτοκόλλων κλειδωμάτων. Στην τελευταία έκδοσή του (έκδοση 6.1 που χρησιμοποιούμε και εμείς), ουσιαστικά υποστηρίζεται μόνο το lock_dlm που είναι ο κατανεμημένος έλεγχος κλειδωμάτων.
- Ένα πλήθος άλλων ρυθμίσεων μπορεί να καθοριστεί σε αυτή τη διαδικασία.

Σε αντίθεση με το GPFS, τα συστήματα αρχείων δεν γίνονται mount αυτοματοποιημένα, αλλά πρέπει κάθε κόμβος να κάνει την διαδικασία ξεχωριστά ανάλογα με τον τύπο της πρόσβασης που έχει στον αποθηκευτικό χώρο.
Για κόμβους με άμεση πρόσβαση στον αποθηκευτικό χώρο, αρκεί να εκτελέσουν την εντολή mount με παραμέτρους τον δίσκο ή το λογικό partition στο οποίο βρίσκεται το σύστημα αρχείων καθώς και τον κατάλογο του συστήματος από τον οποίο θα είναι προσβάσιμο το GFS σύστημα αρχείων.
Για κόμβους που δεν έχουν άμεση πρόσβαση στον αποθηκευτικό χώρο, αλλά χρησιμοποιούν το μοντέλο πρόσβασης NBD τα πράγματα είναι λίγο διαφορετικά. Αρχικά θα πρέπει ο κόμβος που έχει άμεση πρόσβαση στον αποθηκευτικό χώρο να κάνει γνωστό το γεγονός αυτό στους άλλους κόμβους του cluster (μέσω ειδική εντολής του GFS) και στη συνέχεια οι κόμβοι που ενδιαφέρονται να αποκτήσουν την NBD πρόσβαση θα πρέπει να την ρυθμίσουν (πάλι μέσω ειδικής GFS εντολής). Το αποτέλεσμα της ρύθμισης αυτής είναι να δημιουργηθεί σε κάθε κόμβο ένα αρχείο συσκευής που αναπαριστά τον χώρο αποθήκευσης.
Το mount του συστήματος αρχείων γίνεται πάνω στο νέο αυτό αρχείο συσκευής.

Κεφάλαιο 7.
I/O Tracing

7.1 Εισαγωγή

Στο κεφάλαιο αυτό θα περιγράψουμε τους λόγους για τους οποίους γίνεται I/O tracing, την ιεραρχία λογισμικού που εμπλέκεται στο παράλληλο I/O καθώς και τις τεχνικές tracing που δύναται να εφαρμοστούν σε κάθε επίπεδο. Επίσης θα γίνει περιγραφούν τα δύο εργαλεία που χρησιμοποιήθηκαν σε αυτή τη μελέτη, το blktrace και το GPFS Tracing.

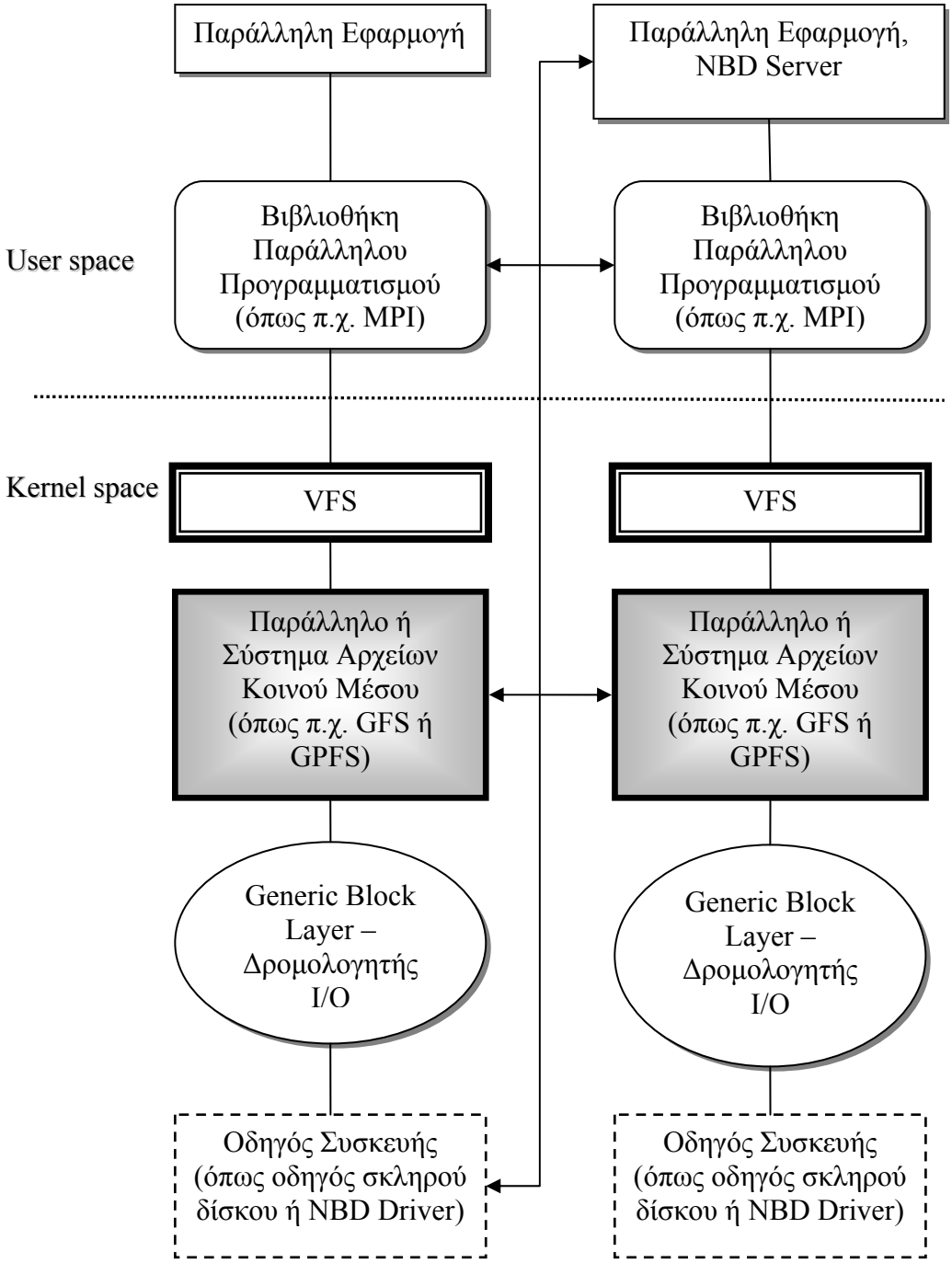
7.2 Λόγοι χρησιμοποίησης I/O tracing

Το I/O tracing είναι η διαδικασία κατά την οποία χρησιμοποιώντας κατάλληλα εργαλεία, συλλέγονται πληροφορίες που αφορούν την I/O κίνηση που δημιουργείται από ορισμένες ή όλες τις εφαρμογές που χρησιμοποιούν τα συστήματα αποθήκευσης ενός υπολογιστή, σε ένα δεδομένο χρονικό διάστημα.
Δύο είναι οι λόγοι που είναι χρήσιμο το I/O tracing.
Πρώτον μπορεί να δώσει πληροφορίες για την I/O συμπεριφορά ορισμένων εφαρμογών. Οι πληροφορίες αυτές ενδεχομένως να είναι πολύ σημαντικές καθώς με σωστή ανάλυσή τους μπορούν να βγουν συμπεράσματα που θα βοηθήσουν στον επανασχεδιασμό των εφαρμογών με στόχο την βελτίωση των I/O επιδόσεών τους.
Επίσης οι πληροφορίες αυτές μπορούν να βοηθήσουν στην διαδικασία του debugging των εφαρμογών αφού με βάση αυτές μπορούν να αναγνωριστούν I/O συμπεριφορές που δεν συνάδουν με την αναμενόμενη I/O συμπεριφορά.
Ο δεύτερος λόγος που χρησιμοποιείται I/O tracing είναι για την απόκτηση γνώσης του τρόπου λειτουργίας και των επιδόσεων των συστημάτων αρχείων και του δρομολογητή I/O του πυρήνα (εφόσον αυτός υπάρχει).

Όπως θα φανεί και στην επόμενη ενότητα, το I/O Tracing μπορεί να εφαρμοστεί σε διαφορετικά επίπεδα της ιεραρχίας λογισμικού για I/O, με διαφορετική δυσκολία υλοποίησης και διαφορετικού είδους εξαγόμενα αποτελέσματα.

7.3 Tracing στην Ιεραρχία Λογισμικού για Παράλληλο I/O

Η ιεραρχία του λογισμικού που εμπλέκεται στο παράλληλο I/O είναι αρκετά παρόμοια με την αντίστοιχη που εμπλέκεται στο I/O μη παράλληλων εφαρμογών. Παρακάτω δίνεται ένα σχήμα αυτής της ιεραρχίας.



Εικόνα 7.1: Ιεραρχία Λογισμικού για Παράλληλο I/O

Στο πρώτο επίπεδο συναντάμε την παράλληλη εφαρμογή. Η εφαρμογή αυτή σχεδιάζεται για να επιλύσει ένα συγκεκριμένο πρόβλημα αξιοποιώντας την εγγενή δυνατότητα παραλληλοποίησης που παρουσιάζει το πρόβλημα.
Συνήθως για την επικοινωνία και τον συντονισμό των εφαρμογών που τρέχουν παράλληλα σε διαφορετικούς υπολογιστές, χρησιμοποιείται μια Βιβλιοθήκη Παράλληλου Προγραμματισμού όπως η βιβλιοθήκη MPI.
Η εφαρμογή ανάλογα με τις ανάγκες που υπάρχουν, μπορεί είτε να κάνει παράλληλο I/O με την χρήση του αντίστοιχου τμήματος της Βιβλιοθήκης Παράλληλου

Προγραμματισμού, είτε τοπικό I/O κάνοντας χρήση των system calls ή βιβλιοθηκών I/O που δημιουργούνται με βάση τα system calls.

Στο δεύτερο επίπεδο συναντάει κανείς το Βιβλιοθήκη Παράλληλου Προγραμματισμού. Η βιβλιοθήκη αυτή εκτός από την υλοποίηση της επικοινωνίας και του συντονισμού ανάμεσα στις παράλληλες εφαρμογές, συνήθως περιλαμβάνει και ένα συγκεκριμένο προγραμματιστικό μοντέλο για παράλληλο I/O.
Όποιες υπηρεσίες και να προσφέρει η βιβλιοθήκη για παράλληλο I/O, η υλοποίησή τους σε χαμηλό επίπεδο θα πρέπει να γίνει με τα γνωστά system calls που αφορούν το σύστημα αρχείων.

Από το τρίτο επίπεδο και κάτω ο κώδικας που υλοποιεί τα επίπεδα εκτελείται σε επίπεδο πυρήνα.

Το τρίτο επίπεδο είναι το VFS Layer. Αυτό το επίπεδο λαμβάνει όλες τις κλήσεις που αφορούν τα συστήματα αρχείων και καλεί τις αντίστοιχες συναρτήσεις του κατάλληλου συστήματος αρχείων.

Στο επόμενο επίπεδο συναντάμε το σύστημα αρχείων. Στην περίπτωση που πολλοί κόμβοι προσπελαύνουν ένα κοινό αποθηκευτικό μέσο χρησιμοποιείται ένα παράλληλο ή ένα σύστημα αρχείων κοινού μέσου. Το σύστημα αυτό εκτός από τις κλασικές εργασίες που έχει να επιτελέσει ως ένα σύστημα αρχείων, πρέπει επίσης να διατηρεί την συνέπεια μέσω της επικοινωνίας όλων των κόμβων που χρησιμοποιούν το κοινό μέσο.

Το παρακάτω επίπεδο είναι το Generic Block Layer και ο Δρομολογητής I/O. Το επίπεδο αυτό δεν εξαρτάται από το σύστημα αρχείων και ο σκοπός του είναι να διαχειρίζεται τις αιτήσεις μεταφοράς block των αποθηκευτικών μέσων που δημιουργεί το σύστημα αρχείων, να τις επεξεργάζεται συνενώνοντας ορισμένες ή δημιουργώντας νέες και στη συνέχεια να τις στέλνει με μια σειρά που το ίδιο επιλέγει, στον οδηγό της αποθηκευτικής συσκευής.

Ο μοναδικός ρόλος του οδηγού συσκευής είναι να μεταφέρει δεδομένα από και προς την συσκευή. Ο οδηγός θα πρέπει να γνωρίζει επακριβώς την μορφή, τον τρόπο λειτουργίας και το πρωτόκολλο επικοινωνίας που χρησιμοποιεί η συσκευή. Παραδείγματα οδηγών συσκευών είναι ο οδηγός για σκληρό δίσκο και ο οδηγός για NBD αποθηκευτικό μέσο. Στην περίπτωση που χρησιμοποιείται NBD οδηγός, θα πρέπει αυτός να επικοινωνεί με την εφαρμογή που τρέχει ως NSD Server στον κόμβο με την άμεση πρόσβαση στον αποθηκευτικό χώρο.

Σε κάθε ένα από αυτά τα επίπεδα λογισμικού μπορεί να γίνει tracing. Τα γεγονότα που μπορούν να καταγραφούν διαφέρουν από επίπεδο σε επίπεδο όπως διαφέρει και η δυσκολία με την οποία υλοποιείται το σύστημα tracing.
Παρακάτω εξετάζουμε την δυνατότητα υλοποίησης tracing για κάθε ένα από τα επίπεδα που αναφέρθηκαν παραπάνω.

Εφαρμογή

Στο επίπεδο της εφαρμογής μπορούμε να καταγράψουμε επακριβώς τις ενέργειες που επιτελεί και συνεπώς να μελετήσουμε την συμπεριφορά της.
Όσον αφορά τις I/O καταγραφές που μας ενδιαφέρουν, μπορούμε να καταγράψουμε το είδος της πρόσβασης που απαιτείται, τα αρχεία στα οποία γίνεται πρόσβαση, το μέγεθος των προς μεταφορά δεδομένων, τις μεταβλητές του προγράμματος οι οποίες λαμβάνουν χώρα στην μεταφορά, τις συνθήκες κάτω από τις οποίες απαιτήθηκε πρόσβαση στον αποθηκευτικό χώρο, τον χρόνο που απαιτήθηκε για να ολοκληρωθούν οι

κλήσεις συναρτήσεων που αφορούν την μεταφορά των δεδομένων και γενικά ότι έχει να κάνει με τον κώδικα της εφαρμογής.
Το είδος αυτό της καταγραφής δεν απαιτεί την ύπαρξη κάποιου εργαλείου, αλλά μόνο τροποποιήσεις στον κώδικα της εφαρμογής ώστε να γίνονται οι καταγραφές όταν εκτελούνται συγκεκριμένα σημεία του κώδικα.
Μια δυσκολία που έχει η συγκεκριμένη μέθοδος είναι ότι πρέπει να γίνει ξεχωριστά για κάθε εφαρμογή που μας ενδιαφέρει. Επιπλέον για εφαρμογές που δεν έχουν ελεύθερο τον πηγαίο κώδικά τους, είναι αδύνατο να εφαρμοστεί.

Βιβλιοθήκη

Όταν γίνεται καταγραφή σε αυτό το επίπεδο, τα δεδομένα που λαμβάνονται είναι μια λίστα από κλήσεις συναρτήσεων της βιβλιοθήκης που κάνει κάθε εφαρμογή μαζί με τις παραμέτρους των κλήσεων αυτών. Επίσης καταγράφονται και οι χρόνοι που αφορούν τις κλήσεις των συναρτήσεων της βιβλιοθήκης και η ποσότητα των δεδομένων που ενδεχομένως μεταφέρονται. Όλα αυτά τα καταγεγραμμένα δεδομένα μπορούν στη συνέχεια να αναλυθούν και να παρουσιαστούν με φιλικό προς τον χρήστη τρόπο, από ειδικά εργαλεία.

Τα πλεονεκτήματα που συνεπάγεται η χρήση αυτής της μεθόδου καταγραφής σε σχέση με την προηγούμενη είναι δύο.
Πρώτον δεν χρειάζεται κάθε εφαρμογή που χρησιμοποιεί την βιβλιοθήκη να κάνει δική της καταγραφεί. Με τροποποίηση στον κώδικα της βιβλιοθήκης και ενδεχομένως μια μικρή τροποποίηση στον κώδικα της εφαρμογής ώστε να δηλώνεται ρητά προς την βιβλιοθήκη ότι είναι απαραίτητη η καταγραφή, μπορούν να καταγραφούν οι κλήσεις για τις συναρτήσεις της βιβλιοθήκης, από όλες τις εφαρμογές που την χρησιμοποιούν. Με αυτό τον τρόπο γλυτώνετε πολύς χρόνος από την ανάπτυξη συστήματος καταγραφής για κάθε εφαρμογή που θα χρησιμοποιήσει την βιβλιοθήκη.
Δεύτερον υπάρχουν πολλά εργαλεία που μπορούν να κάνουν λεπτομερή ανάλυση των καταγεγραμμένων δεδομένων των οποίων η χρήση τους σε αυτό το επίπεδο δεν απαιτεί καμία επιπλέον προσπάθεια αφού τα καταγεγραμμένα δεδομένα που λαμβάνουν ως είσοδο είναι σε κατάλληλη μορφή. Αντίθετα αν ήθελε κανείς να χρησιμοποιήσει αυτά τα εργαλεία με καταγραφές που έγιναν στο επίπεδο της εφαρμογής, θα έπρεπε ο προγραμματιστής που ανάπτυξε το σύστημα καταγραφής των ενεργειών της εφαρμογής, να εξάγει τα δεδομένα σε μορφή που είναι συμβατή με το πρόγραμμα ανάλυσης των δεδομένων.

Στο επίπεδο αυτό της βιβλιοθήκης, υπάρχουν πολλά εργαλεία που εισάγουν τροποποιήσεις στην βιβλιοθήκη MPI ώστε να είναι δυνατή η καταγραφή των I/O (όπως επίσης και άλλων) γεγονότων και στη συνέχεια αναλύουν τις καταγραφές ώστε οι προγραμματιστές των MPI εφαρμογών να μπορούν να δουν επακριβώς την συμπεριφορά τους και να εντοπίζουν bottlenecks.
Τα πιο σημαντικά από αυτά τα εργαλεία είναι τα: DEEP/ MPI, MPE Logging and Jumpshot, Pablo Perfomance Analysis Tools, Paradin, Vampir, TAU, LANL-Trace και Intel Trace Analyzer and Collector.

Strace

Στο επίπεδο της εφαρμογής υπάρχει η δυνατότητα να καταγράφονται και τα system calls που κάνει κάθε εφαρμογή.
Κάθε εφαρμογή είτε χρησιμοποιεί ειδικές βιβλιοθήκες ή όχι, για την χαμηλού επιπέδου υλοποίηση της πρόσβασης στον αποθηκευτικό χώρο θα χρησιμοποιεί αναγκαστικά τα system calls που διαθέτει για αυτό τον λόγο το λειτουργικό σύστημα. Συνεπώς τα system calls πάντα «προδίδουν» την συμπεριφορά της εφαρμογής όσον αφορά το I/O. Υπάρχει μόνο μία εξαίρεση σε αυτόν τον κανόνα. Υπάρχουν ορισμένες εφαρμογές που χρησιμοποιούν memory mapped I/O και αποφεύγουν την χρήση των ειδικών για αρχεία system calls. Σε αυτή την περίπτωση η προσπέλαση στα δεδομένα του αρχείου υλοποιείται απλά με ανάγνωση και εγγραφή της μνήμης της εφαρμογής και δεν απαιτούνται system calls.

Η δυνατότητα καταγραφής των system calls απαιτεί υποστήριξη από τον πυρήνα του λειτουργικού συστήματος και την ύπαρξη ενός προγράμματος επιπέδου χρήστη που θα διαβάζει τα δεδομένα που κατέγραψε ο πυρήνας και θα τα παρουσιάζει στον χρήστη. Ο πυρήνας του λειτουργικού συστήματος Linux υποστηρίζει την καταγραφή των system calls χωρίς να απαιτείται κάποιο patch, αφού όλη η απαραίτητη υποδομή υπάρχει στον κώδικά του, ενώ για την παρουσίαση των δεδομένων των καταγραφών χρησιμοποιείται το πρόγραμμα Strace.
Το Strace μπορεί να καταγράψει οποιοδήποτε system call κάνει μια εφαρμογή αλλά επίσης μπορεί με την χρήση φίλτρων να κάνει καταγραφή μόνο συγκεκριμένων system call. Στην περίπτωση που απαιτείται καταγραφή I/O γεγονότων το Strace μπορεί να καταγράψει μόνο τα system calls που σχετίζονται με το σύστημα αρχείων. Τα δεδομένα που καταγράφονται σε αυτή την περίπτωση είναι μια λίστα από system calls και παραμέτρους με τις οποίες έγιναν καθώς και τους χρόνους που κλήθηκαν και ολοκληρώθηκαν.

Το μεγάλο πλεονέκτημα που έχει η χρήση του Strace είναι ότι δεν απαιτείται απολύτως καμία τροποποίηση της προς εξέταση εφαρμογής. Έτσι είναι καθίσταται εφικτή η δυνατότητα να καταγραφεί η I/O συμπεριφορά όλων των εφαρμογών, συμπεριλαμβανομένων αυτών που υπάρχουν μόνο σε εκτελέσιμη μορφή.

Οι μέθοδοι που εξετάστηκαν παραπάνω λειτουργούν στο επίπεδο του χρήστη και συνεπώς όλες οι καταγραφές γίνονται σε αυτό το επίπεδο. Αυτό συνεπάγεται ότι η αποθήκευση των καταγραφών μπορεί να γίνει πολύ απλά σε αρχεία με χρήση των system calls.
Οι επόμενες μέθοδοι καταγραφής λειτουργούν στο επίπεδο του πυρήνα. Εξ αιτίας αυτού του γεγονότος, για την υλοποίησή τους απαιτείται επιπλέον προσπάθεια, εφόσον δεν διατίθεται το απλό προγραμματιστικό interface του επιπέδου χρήστη.

VFS

Αν και η καταγραφή δεδομένων σε αυτό το επίπεδο είναι δυνατή τεχνικά (τροποποιώντας τον κώδικα του πυρήνα ώστε να υποστηρίζεται αυτή η δυνατότητα), εντούτοις δεν προσφέρει κάποιο πλεονέκτημα σε σχέση με την προηγούμενη μέθοδο και ως εκ τούτου δεν έχει εφαρμοστεί στο Linux.

Η μόνη επιπλέον από το Strace πληροφορία που μπορεί να δώσει, είναι το συγκεκριμένο σύστημα αρχείων που περιέχει τα προς προσπέλαση δεδομένα.

Πρέπει να δώσουμε έμφαση στο γεγονός ότι οι μέθοδοι καταγραφών που εξετάστηκαν παραπάνω, δίνουν πληροφορίες για την συμπεριφορά των εφαρμογών μόνο. Δεν μπορούν να δώσουν πληροφορίες σχετικά με το πώς δουλεύει στην πράξη το I/O.
Για να πάρει κανείς τέτοιες πληροφορίες θα πρέπει να χρησιμοποιήσει μια από τις παρακάτω μεθόδους.

File System

Η καταγραφή σε αυτό το επίπεδο μπορεί να δώσει πάρα πολλές πληροφορίες σχετικά με το πώς γίνεται ο χειρισμός των I/O αιτήσεων και η διαχείριση του αποθηκευτικού χώρου.
Για κάθε αίτηση πρόσβασης σε κάποιο αρχείο που δημιουργείται από μια εφαρμογή και περνάει στο σύστημα αρχείων, η καταγραφή μπορεί να μας δώσει τις παρακάτω πληροφορίες:

- Τους προσδιοριστικούς αριθμούς των δομών δεδομένων που αναπαριστούν το αρχείο.
  Ένα χρήσιμο συμπέρασμα που θα μπορούσαμε να βγάλουμε χρησιμοποιώντας μια τέτοια πληροφορία είναι να δούμε αν ένα αρχείο που αποτελεί softlink σε ένα άλλο, αναπαρίσταται από το ίδιο inode ή όχι.
- Αν η αίτηση αφορά την δημιουργία ενός νέου αρχείου, η καταγραφή μπορεί να πληροφορήσει για τα φυσικά blocks του δίσκου που δεσμεύονται για τα δεδομένα του αρχείου.
- Αν το αρχείο διαγράφεται παίρνουμε πληροφορίες για τα blocks του δίσκου που αποδεσμεύονται και θεωρούνται πλέον ελεύθερος χώρος.
- Αν η αίτηση αφορά την ανάγνωση ή την εγγραφή δεδομένων του αρχείου, μπορούμε να πάρουμε πληροφορίες που αφορούν τους χρόνους έναρξης και ολοκλήρωσης της μεταφοράς όλων των προς μεταφορά φυσικών blocks του δίσκου που αντιστοιχούν στα συγκεκριμένα δεδομένα του αρχείου.

Επιπλέον οι καταγραφές σε αυτό το επίπεδο μπορούν να δώσουν πληροφορίες για την ποσότητα της κεντρικής μνήμης που χρησιμοποιεί το σύστημα αρχείων, την αλληλεπίδρασή του με τον υπόλοιπο πυρήνα, την ποσότητα του αποθηκευτικού χώρου που δεσμεύεται για τις ανάγκες του συστήματος αρχείων καθώς και διάφορα στατιστικά που αφορούν τις επιδόσεις ανάγνωσης και εγγραφής.

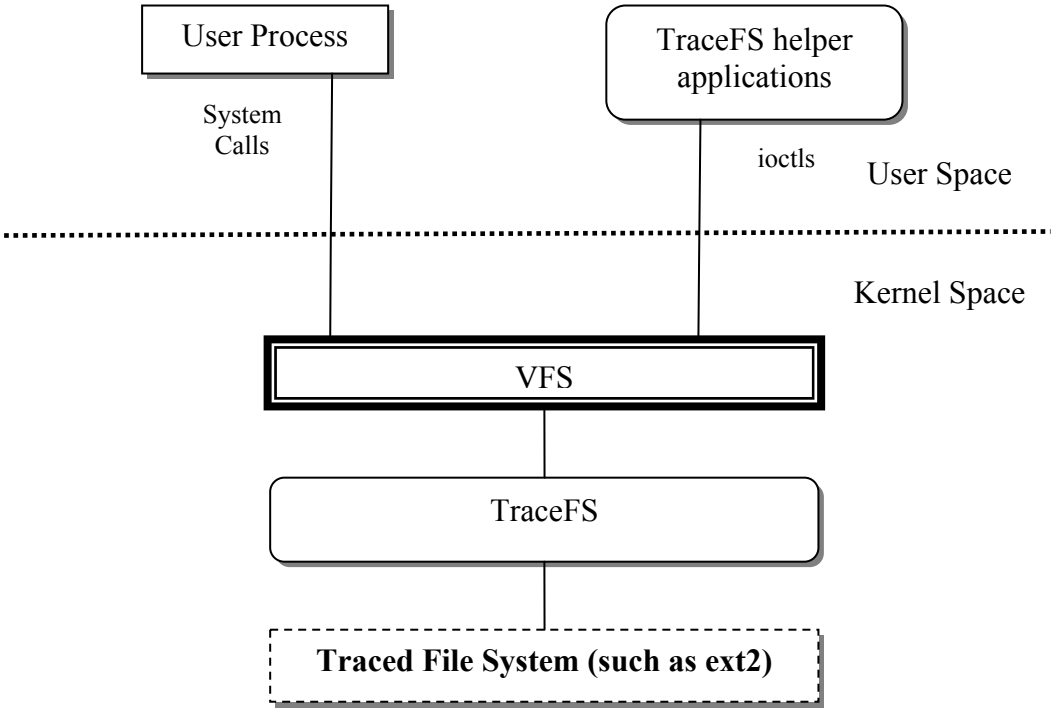
Στην περίπτωση του παράλληλου I/O χρησιμοποιείται ένα παράλληλο ή ένα σύστημα αρχείων κοινού μέσου. Σε αυτά τα συστήματα αρχείων είναι δυνατόν σε κάθε αίτηση για πρόσβαση σε αρχεία του συστήματος αρχείων, να καταγράφονται πληροφορίες για το είδος και το πλήθος των κλειδωμάτων που απαιτήθηκαν καθώς και πληροφορίες για ενδεχόμενη προώθηση της αίτησης σε άλλους κόμβους που χρησιμοποιούν το σύστημα αρχείων.

Παρά την σπουδαιότητα του συστήματος αυτού, πολύ λίγα συστήματα αρχείων το διαθέτουν. Ένα από αυτά είναι το GPFS ενώ κάποια άλλα είναι το σύστημα αρχείων

του 4.2 BSD και το Sprite.

TraceFS

Ένας τρόπος για να ξεπεραστεί η απουσία συστήματος καταγραφής στο επίπεδο του συστήματος αρχείων είναι η χρήση του TraceFS. Το TraceFS αποτελεί ένα κομμάτι λογισμικού που αναπτύχθηκε προκειμένου να είναι δυνατή η πραγματοποίηση tracing σε οποιοδήποτε σύστημα αρχείων. Υλοποιείται ως ένα επιπλέον επίπεδο της ιεραρχίας λογισμικού για I/O και παρεμβάλλεται ανάμεσα στο VFS και το πραγματικό σύστημα αρχείων όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 7.2: TraceFS

Λόγω της σχεδίασής του το TraceFS μπορεί να χρησιμοποιηθεί για tracing οποιουδήποτε συστήματος αρχείων. Αυτό είναι εφικτό καθώς καταγράφει τις κλήσεις που κάνει το VFS προς τα συστήματα αρχείων και τα αποτελέσματα που επιστρέφουν τα συστήματα αρχείων ως αποτέλεσμα των κλήσεων αυτών. Με αυτό τον τρόπο δύναται να καταγράψει τις πληροφορίες που αναφέρθηκαν στο τμήμα του συστήματος αρχείων.

Στην μελέτη μας θα είχε ενδιαφέρον να χρησιμοποιήσουμε το TraceFS προκειμένου να δούμε τι επιπλέον συμπεράσματα θα μπορούσαμε να βγάλουμε, αλλά δυστυχώς αυτό δεν ήταν δυνατό για τεχνικούς λόγους. Πιο συγκεκριμένα το TraceFS δημιουργήθηκε για την έκδοση 2.4 του πυρήνα του Linux και εκ τότε δεν έχει ανασχεδιαστεί για την έκδοση 2.6 που είναι αναγκαία για την λειτουργία του GFS και του GPFS που μελετάμε.

Generic Block Layer – I/O Scheduler

Στο επίπεδο αυτό οι πληροφορίες που παίρνουμε από το tracing αφορούν την πορεία που ακολουθούν οι αιτήσεις μεταφοράς blocks των αποθηκευτικών συσκευών στην ουρά αναμονής που δημιουργεί ο πυρήνας για κάθε τέτοια συσκευή. Μπορούμε δηλαδή να πάρουμε πληροφορίες σχετικά με την δημιουργία των αιτήσεων, την διάσπασή τους, την συνένωσή τους με άλλες, την αποστολή τους προς τον αντίστοιχο οδηγό (driver) της συσκευής και την ολοκλήρωσή τους.
Για να είναι δυνατό το tracing σε αυτό το επίπεδο, θα πρέπει το λειτουργικό σύστημα να ορίζει και να χρησιμοποιεί αυτό το επίπεδο. Η αναφορά αυτή γίνεται επειδή το επίπεδο αυτό δεν είναι σε καμία περίπτωση απαραίτητο για την διεκπαιρέωση του I/O (είναι όμως απαραίτητο για επίτευξη ικανοποιητικών I/O επιδόσεων) και σε κάποια (είτε παλαιά είτε μικρά ειδικού σκοπού) λειτουργικά συστήματα οι αιτήσεις μεταφοράς blocks που δημιουργεί το σύστημα αρχείων στέλνονται κατευθείαν στον οδηγό της συσκευής, χωρίς να υφίστανται επεξεργασία από το επίπεδο που συζητάμε.

Για να είναι δυνατή η καταγραφή σε αυτό το επίπεδο απαιτείται υποστήριξη από τον πυρήνα. Στο Linux αυτή η υποστήριξη υπάρχει με την μορφή patch για τους πυρήνες 2.6.14 και 2.6.16 ενώ από την έκδοση 2.6.17 και μετά υποστηρίζεται εγγενώς.

Drivers

Το tracing σε αυτό το επίπεδο δίνει πληροφορίες σχετικά με το πώς ο οδηγός μιας συσκευής διαχειρίζεται τις αιτήσεις πρόσβασης στην συσκευή αυτή. Οι σημαντικότερες τέτοιες πληροφορίες είναι οι συνθήκες κάτω από τις οποίες ο οδηγός της συσκευής εκκινεί και ολοκληρώνει την μεταφορά των δεδομένων.
Για να είναι δυνατό αυτό το είδος του tracing, θα πρέπει ο κώδικας που υλοποιεί τον οδηγό της συσκευής, να παρέχει την υποδομή ώστε να γίνεται η καταγραφή των απαραίτητων δεδομένων αλλά και την δυνατότητα να εξάγονται τα δεδομένα αυτά στο επίπεδο του χρήστη.
Το μειονέκτημα που έχει αυτή η μέθοδος tracing σε σύγκριση με την προηγούμενη, είναι ότι για να είναι δυνατή η καταγραφή δεδομένων σε κάθε αποθηκευτική συσκευή που μας ενδιαφέρει, θα πρέπει να τροποποιηθούν όλοι οι αντίστοιχοι οδηγοί σε αντίθεση με την προηγούμενη μέθοδο όπου απαιτείται τροποποίηση σε ένα μόνο κομμάτι κώδικα του πυρήνα.

7.4 blktrace

Το blktrace είναι ένα μηχανισμός καταγραφής I/O κίνησης στο επίπεδο του δρομολογητή I/O του Linux. Ουσιαστικά αποτελεί ένα εργαλείο tracing. Παρέχει στο επίπεδο χρήστη λεπτομερείς πληροφορίες σχετικά με την κατάσταση της ουράς αναμονής των I/O αιτήσεων κάποιας συσκευής που επιλέγει ο χρήστης.

Το blktrace αποτελείται από τρία υποσυστήματα:

1. Ένα κομμάτι κώδικα στον πυρήνα του Linux που αναλαμβάνει την καθ' αυτό καταγραφή της I/O κίνησης καθώς αυτή υφίσταται επεξεργασία μέσα στην ουρά αναμονής κάθε συσκευής.

2. Ένα κομμάτι που αναλαμβάνει να μεταφέρει τα καταγεγραμμένα δεδομένα από τον πυρήνα στο επίπεδο του χρήστη.
3. Ένα κομμάτι που επεξεργάζεται τα δεδομένα και τα εξάγει σε κατανοήσιμη μορφή.

Το είδος των γεγονότων που μπορεί να καταγράψει το blktrace είναι: Η δημιουργία αιτήσεων πρόσβασης στον δίσκο, η συνένωση πολλών αιτήσεων που αφορούν blocks δεδομένων που βρίσκονται σε διπλανές θέσεις στην συσκευή αποθήκευσης σε μία, η διάσπαση μιας αίτησης σε πολλές μικρότερες, η αποστολή της αίτησης προς τον οδηγό (driver) της συσκευής αποθήκευσης και τέλος την εξυπηρέτηση της αίτησης από την συσκευή.
Οι συγκεκριμένες πληροφορίες που παίρνουμε από το blktrace για κάθε καταγεγραμμένο γεγονός, είναι το αρχικό block του δίσκου και το πλήθος των επόμενων block που λαμβάνουν μέρος στην μεταφορά, τον αριθμό PID της διεργασίας που οδήγησε στην δημιουργία του καταγεγραμμένου γεγονότος, τον επεξεργαστή στον οποίο έτρεχε η διεργασία, τους major και minor αριθμούς του οδηγού που εξυπηρετεί την συσκευή στην οποία θα γίνει προσπέλαση, τον χρόνο που συνέβη το γεγονός (σε σχέση με τον χρόνο 0 που θεωρείται ο χρόνος κατά τον οποίο κατάγράφηκε το πρώτο γεγονός) και τον αύξοντα αριθμό του γεγονότος.
Όταν ο χρήστης επιλέξει να σταματήσει την καταγραφή, το blktrace του παρουσιάζει ορισμένα συνοπτικά δεδομένα που αφορούν το πλήθος των δεδομένων που διαβάστηκαν και γράφτηκαν από και προς την συσκευή αποθήκευσης αντίστοιχα, τον μέσο ρυθμό διαμεταγωγής με τον οποίο έγιναν αυτές οι ενέργειες καθώς και τον αριθμό των αιτήσεων που συγχωνεύτηκαν, διασπάστηκαν και ολοκληρώθηκαν.

Ορισμένες λεπτομέρειες που αφορούν τον τρόπο λειτουργία του blktrace είναι οι εξής:

Η μεταφορά των καταγεγραμμένων δεδομένων από τον πυρήνα στο blktrace που βρίσκεται στο επίπεδο χρήστη γίνεται μέσω του ειδικού συστήματος αρχείων Relayfs για πυρήνες παλαιότερους από 2.6.17 ή του Debugfs για πυρήνες από 2.6.17 και νεότερους. Τα δεδομένα της καταγραφής καταλήγουν σε buffers του blktrace. Το μέγεθος και το πλήθος των buffers αυτών είναι παραμετροποιήσιμα μέσω του blktrace μεγέθη.

Το blktrace δίνει την δυνατότητα στον χρήστη να επιλέξει ορισμένα φίλτρα προκειμένου τα δεδομένα καταγραφής που θα πάρει να είναι μόνο αυτά που τον ενδιαφέρουν. Τα φίλτρα αυτά είναι φίλτρα απεικόνισης και όχι καταγραφής καθώς η καταγραφή γίνεται στο επίπεδο του πυρήνα και δεν υπάρχει τρόπος παραμετροποίησής της.

Για την καλύτερη οπτικοποίηση των δεδομένων που προκύπτουν από το blktrace συνίσταται η χρήση του προγράμματος Blkparse. Η τυπική χρήση του γίνεται με την χρήση pipe από το blktrace προς το Blkparse.

Εμείς στην περίπτωσή μας θα χρησιμοποιήσουμε το blktrace προκειμένου να βγάλουμε συμπεράσματα κυρίως για το πως χειρίζονται θέματα caching τα συστήματα αρχείων GFS και GPFS.
Θα χρησιμοποιήσουμε κάποια ειδικά σενάρια όπου διάφοροι κόμβοι θα προσπελαύνουν το σύστημα αρχείων και από τις καταγραφές που θα μας δίνει το

blktrace θα μπορέσουμε να δούμε ποιοτικά αλλά και ποσοτικά πως τα συστήματα αυτά αρχείων χρησιμοποιούν την cache. Το blktrace μπορεί να μας δώσει πληροφορίες που αφορούν το caching των δεδομένων έμμεσα, παρατηρώντας την ανυπαρξία προσβάσεων στον δίσκο ενώ ζητήθηκαν δεδομένα του συστήματος αρχείων.
Για παράδειγμα αν ένας κόμβος διαβάσει ένα ολόκληρο αρχείο και από την καταγραφή του δούμε πως από τον δίσκο μεταφέρθηκαν μόνο τα μισά δεδομένα του αρχείου, συμπεραίνουμε πως τα υπόλοιπα δεδομένα βρίσκονταν στην cache.

7.5 GPFS Tracing

Όταν χρησιμοποιούμε το GPFS μπορούμε να χρησιμοποιήσουμε τις δυνατότητες καταγραφής του ιδίου του GPFS αντί για το blktrace.
Η περιγραφεί του tracing στην ιεραρχία του λογισμικού που εμπλέκεται στο παράλληλο I/O, συνεπάγεται ότι το tracing που κάνει το GPFS είναι ένα επίπεδο πάνω από το επίπεδο του tracing του blktrace, δηλαδή βρίσκεται στο επίπεδο του συστήματος αρχείων. Συνεπώς δεν έχει την δυνατότητα να καταγράψει γεγονότα που αφορούν την διαχείριση των αιτήσεων στην ουρά αναμονής κάθε συσκευής, αλλά μπορεί να καταγράψει όλες τις λειτουργίες που επιτελεί το σύστημα αρχείων.
Πέρα από την δυνατότητα καταγραφής διαφόρων γεγονότων, το GPFS επίσης δίνει την δυνατότητα του ελέγχου της λεπτομέρειας των καταγεγραμμένων δεδομένων.

Τα γεγονότα που μπορούν να καταγραφούν με την χρήση του GPFS tracing είναι γεγονότα που αφορούν:

- Διαχείριση ελεύθερου αποθηκευτικού χώρου
- Προσπέλαση στον αποθηκευτικό χώρο
- Διαχείριση κλειδωμάτων
- Data Shipping
- Διαχείριση του cluster και του συστήματος αρχείων
- Επιδόσεις του συστήματος αρχείων
- Αλληλεπίδραση με τον πυρήνα του Linux

Κεφάλαιο 8. Πειραματικά Συμπεράσματα για τα GFS και GPFS

8.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα περιγράψουμε με λεπτομέρεια τα πειράματα και τα συμπεράσματα που προέκυψαν από την διεξαγωγή των πειραμάτων αυτών. Επιπλέον θα γίνει περιγραφή του λογισμικού που χρησιμοποιήθηκε για την διεξαγωγή των πειραμάτων.

8.2 Λογισμικό

Για την διεξαγωγή των πειραμάτων χρησιμοποιήθηκαν ορισμένα επιπλέον προγράμματα, πέραν των GFS, GPFS και του blktrace.
Το λογισμικό αυτό χωρίζεται σε δύο κατηγορίες με βάση το σκοπό του.
Στην πρώτη κατηγορία ανήκουν τα προγράμματα που προσπελαύνουν αρχεία με συγκεκριμένο τρόπο.
Στην δεύτερη κατηγορία ανήκουν προγράμματα που συγκρίνουν διαφορετικές καταγραφές και αποφαίνονται για την ομοιότητα ή διαφορετικότητά τους. Δηλαδή αποφαίνονται για αν δύο καταγραφές περιέχουν τα ίδια προσπελασθέντα blocks δεδομένων ή όχι.

8.2.1 mBench

Το πρόγραμμα αυτό αναπτύχθηκε από τον γράφοντα με σκοπό να είναι ένα benchmark συγκεκριμένου σκοπού το οποίο θα επιτρέπει σε έναν κόμβο να κάνει αναγνώσεις που θα βοηθήσουν στην εξαγωγή συγκεκριμένων συμπερασμάτων για τα δύο υπό μελέτη συστήματα αρχείων.
Οι δυνατότητες που διαθέτει το mBench είναι οι εξής:

- Κάνει ανάγνωση οποιουδήποτε αρχείου χρησιμοποιώντας sequential, reverse sequential, random και constant access patterns.
- Για sequential και reverse patterns, μπορεί να διαβάσει αντί για όλο το αρχείο, μόνο συγκεκριμένα τέταρτα ή μισά του.
- Για random και constant access patterns μπορεί να καθοριστεί το πλήθος των αναγνώσεων που θα γίνουν.
- Το μέγεθος των δεδομένων που θα αναγνωστούν σε κάθε read system call είναι μεταβλητό και μπορεί να καθοριστεί από τον χρήστη.
- Έχει την δυνατότητα καταγραφής του χρόνου που απαιτήθηκε για κάθε μια ανάγνωση αλλά και τον συνολικό χρόνο που έτρεξε.

Για κάποια από τα πειράματα που έγιναν και θα περιγραφούν αργότερα, δεν υπήρχαν έτοιμα benchmark που θα μπορούσαν να κάνουν την απαραίτητη δουλειά, συνεπώς η

ανάπτυξη ενός ειδικού σκοπού benchmark, όπως το mBench ήταν απαραίτητη.
Το πρόγραμμα αυτό αναπτύχθηκε στην γλώσσα C και χρησιμοποιεί τα γνωστά UNIX system calls προκειμένου να επιτελέσει τις διάφορες ενέργειές του.

8.2.2 mWrite

Το πρόγραμμα mWrite δημιουργήθηκε από τον γράφοντα προκειμένου να κάνει πολύ απλή την διαδικασία της εγγραφή σε οποιοδήποτε αρχείο από οποιονδήποτε κόμβο.
Σε ορισμένα πειράματα που θα δούμε παρακάτω, η τροποποίηση κάποιου αρχείου προηγείται των επόμενων βημάτων του πειράματος. Με χρήση του mWrite είναι δυνατή η εύκολη τροποποίηση οποιουδήποτε μέρους του αρχείου (αφού γίνεται σε εγγραφή σε οποιοδήποτε σημείο του αρχείου) ή ακόμα και η επέκτασή του, με γράψιμο στο τέλος του.
Το πρόγραμμα αυτό επίσης αναπτύχθηκε σε C και κάνει χρήση των UNIX system calls για να επιτελέσει τις διάφορες ενέργειές του.

8.2.3 gpfsperf

Το πρόγραμμα αυτό είναι ένα benchmark που παρέχεται με την εγκατάσταση του GPFS για την μελέτη των επιδόσεων του συγκεκριμένου συστήματος αρχείων.
Υποστηρίζει την ταυτόχρονη προσπέλαση στα αρχεία από διαφορετικούς κόμβους με χρήση της βιβλιοθήκης MPI. Η προσπέλαση στα αρχεία μπορεί να γίνει με διάφορους τρόπους.
Ο λόγος για τον οποίο χρησιμοποιήσαμε το gpfsperf είναι ότι παρέχει την δυνατότητα ανάγνωσης και εγγραφής αρχείων με χρήση του Data Shipping Mode του GPFS. Με αυτό τον τρόπο μπορούμε να δούμε το Data Shipping Mode να λειτουργεί στην πράξη.

8.2.4 BTIO

Το BTIO είναι ένα παράλληλο benchmark που κάνει χρήση της βιβλιοθήκης MPI προκειμένου οι κόμβοι που το τρέχουν να επιλύσουν ένα συγκεκριμένο επιστημονικό πρόβλημα και να γράψουν τα αποτελέσματα της λύση αυτής σε ένα κοινό απόθηκευτικό μέσο με τη μορφή ενός ή περισσοτέρων αρχείων. Αφού όλοι οι κόμβοι γράψουν τα αποτελέσματά της επεξεργασίας τους στο αποθηκευτικό μέσο, στη συνέχεια πραγματοποιούν ανάγνωσή των δεδομένων αυτών από το αποθηκευτικό μέσο.
Τα υπό επεξεργασία δεδομένα του προβλήματος αναπαριστά ένας τρισδιάστατος πίνακας. Το BTIO χρησιμοποιεί ένα σχήμα block-tridiagonal (BT) καταμερισμού της επεξεργασίας των δεδομένων του τρισδιάστατου πίνακα στους κόμβους που επιλύουν το πρόβλημα. Μπορούν να χρησιμοποιηθούν ένας, τέσσερις, εννιά ή δεκαέξι κόμβοι για την επίλυση του προβλήματος και κάθε κόμβος είναι υπεύθυνος για την επεξεργασία πολλαπλών τμημάτων των συνολικών δεδομένων.
Το μέγεθος του τρισδιάστατου πίνακα καθώς και το πλήθος των επαναλήψεων που γίνονται πάνω σε αυτόν δεν είναι δεδομένα, αλλά καθορίζονται στον χρόνο μεταγλώττισης, από την κλάση του προβλήματος που επιλέγεται. Οι κλάσεις του

προβλήματος που διατίθενται είναι σε αύξουσα σειρά μεγέθους δεδομένων: S (Sample), W (Workstation), A, B, C, D και E. Σε κάθε κλάση προβλήματος το BTIO παράγει και διαφορετικό μέγεθος αρχείου (ή αρχείων) ως έξοδο.

Η σημαντικότερη λειτουργία που επιτελεί το BTIO δεν είναι η επεξεργασία των δεδομένων, αλλά η εγγραφή των αποτελεσμάτων της επεξεργασίας, στο κοινό αποθηκευτικό μέσο. Ως εκ τούτου το BTIO δίνει την δυνατότητα να εγγραφούν αυτά τα δεδομένα με πολλούς τρόπους. Αυτοί είναι:

- Full MPI I/O. Με αυτό τον τρόπο οι κόμβοι που επιλύουν το πρόβλημα και έχουν δεδομένα προς εγγραφή στο κοινό αποθηκευτικό μέσο τα γράφουν σε ένα μόνο αρχείο, αλλά συνεργάζονται ώστε να γίνεται η εγγραφή γειτονικών blocks από έναν κόμβο. Θεωρητικά με αυτό τον τρόπο μειώνεται η εγγραφή σε ίδια blocks από τους κόμβους.
- Simple MPI I/O. Με τον τρόπο δεν υπάρχει συνεργασία των κόμβων για την εγγραφή των δεδομένων στο κοινό αποθηκευτικό μέσο, αλλά απλά γράφουν τα δεδομένα τους σε ένα κοινό αρχείο χρησιμοποιώντας το I/O κομμάτι της βιβλιοθήκης MPI για να γίνουν οι εγγραφές. Επειδή χρησιμοποιείται ένα μόνο αρχείο και γράφονται από τους κόμβους ίδια blocks, ο τρόπος αυτός εγγραφής θεωρητικά πρέπει να έχει ως αποτέλεσμα άσχημες επιδόσεις όταν χρησιμοποιείται ένα σύστημα αρχείων κοινού μέσου.
- Fortran I/O. Όταν χρησιμοποιείται αυτός ο τρόπος I/O, κάθε κόμβος γράφει στο ίδιο αρχείο τα αποτελέσματά του, αλλά χρησιμοποιώντας Fortran συναρτήσεις.
- Epio. Με αυτή την μέθοδο κάθε κόμβος γράφει τα δεδομένα του σε δικό του αρχείο, χρησιμοποιώντας Fortran I/O.

Όταν χρησιμοποιούνται οι δύο πρώτοι τρόποι προσπέλασης των δεδομένων, το BTIO καθίσταται ένας καλός μετρητή των επιδόσεων του I/O τμήματος της MPI βιβλιοθήκης και του συστήματος αρχείων, καθώς ο τρόπος εγγραφής των δεδομένων είναι ιδιαίτερα απαιτητικός. Ο λόγος είναι ότι όλοι κόμβοι γράφουν μαζί και διαδοχικά στα ίδια μικρά κομμάτια του αρχείου.

Πρέπει να σημειώσουμε ότι το BTIO είναι μέρος της συλλογής benchmark, NAS που δημιουργήθηκε από το NASA Ames Research Center. Η γλώσσα προγραμματισμού με την οποία είναι γραμμένο είναι η Fortran 77. Η έκδοση του BTIO που χρησιμοποιήθηκε στην περίπτωσή μας ήταν η έκδοση 3.3 που αποτελούσε την πλέον πρόσφατη την χρονική περίοδο διεξαγωγής των πειραμάτων.

8.2.5 Προγράμματα comparer

Τα προγράμματα αυτά αναπτύχθηκαν από τον γράφοντα με σκοπό να είναι εύκολη και γρήγορη η αυτοματοποιημένη σύγκριση διαφορετικών καταγραφών που περιέχουν blocks δεδομένων που προσπελάστηκαν από τους κόμβους στους οποίους έγιναν τα πειράματα.
Όπως θα φανεί παρακάτω, ύστερα από την εκτέλεση ορισμένων πειραμάτων προκύπτει η ανάγκη να δούμε αν δύο καταγραφές περιλαμβάνουν κοινά blocks δεδομένων ή όχι. Η μη αυτοματοποιημένη σύγκρισή τους δεν είναι δυνατή πρακτικά καθώς κάθε αρχείο μπορεί να περιλαμβάνει την ανάγνωση δεκάδων χιλιάδων blocks.
Για την ανάπτυξη των προγραμμάτων αυτών χρησιμοποιήθηκε η γλώσσα

προγραμματισμού Java.

8.3 Πειράματα και Συμπεράσματα

Σκοπός των πειραμάτων που διεξήχθησαν ήταν να μπορέσουμε να εξακριβώσουμε στην πράξη, ορισμένα στοιχεία για τον τρόπο λειτουργίας των συστημάτων αρχείων GFS και GPFS. Έμφαση δίνεται κυρίως στην εξακρίβωση του τρόπου χειρισμού του caching των δεδομένων που κάνουν τα δύο αυτά συστήματα αρχείων.
Με το τελευταίο πείραμα σκοπό έχουμε να συγκρίνουμε τους διαφορετικούς τρόπους πρόσβασης στα δεδομένα που μπορεί να χρησιμοποιήσει η εφαρμογή BTIO και να δούμε στην πράξη το Data Shipping Mode του GPFS.

Για την διεξαγωγή των πειραμάτων εγκαταστάθηκαν τα συστήματα αρχείων GFS και GPFS σε εικονικό hardware όπως περιγράφεται στο κεφάλαιο 6. Σημειώνουμε εδώ ότι τα πειράματα διεξήχθησαν για τα δύο συστήματα αρχείων και στην περίπτωση όπου όλοι οι κόμβοι είχαν απευθείας πρόσβαση στους (εικονικούς) δίσκους και στην περίπτωση όπου ένας κόμβος έχει απευθείας πρόσβαση στον δίσκο ενώ οι άλλοι χρησιμοποιούσαν το NBD μοντέλο πρόσβασης. Τα αποτελέσματα των πειραμάτων ήταν ακριβώς τα ίδια για τις δύο περιπτώσεις γι' αυτό και παρακάτω αναφέρονται τα αποτελέσματα μόνο των πειραμάτων όπου όλοι οι κόμβοι είχαν απευθείας πρόσβαση στον κοινό αποθηκευτικό χώρο.
Το λογισμικό που χρησιμοποιήθηκε για την πραγματοποίηση των καταγραφών είναι το blktrace και το GPFS Trace που περιγράφονται στις ενότητες 7.4 και 7.5 αντίστοιχα. Σημειώνουμε ότι στις περιπτώσεις που χρησιμοποιήθηκε το NBD μοντέλο πρόσβασης για το GPFS (που το GPFS ονομάζει NSD Server), δεν ήταν δυνατή η χρήση του blktrace, καθώς το GPFS δεν εξάγει στον κατάλογο /dev την εικονική NBD συσκευή.
Τα προγράμματα που χρησιμοποιήθηκαν για την προσπέλαση στα αρχεία των συστημάτων αρχείων περιγράφονται στην ενότητα 8.2.
Για την διεξαγωγή του τελευταίου πειράματος απαιτήθηκε η εγκατάσταση μιας υλοποίησης της βιβλιοθήκης MPI. Χρησιμοποιήθηκε η MPICH που αποτελεί την πιο κοινά χρησιμοποιούμενη υλοποίηση του MPI. Η συγκεκριμένη έκδοση της που χρησιμοποιήθηκε ήταν η 1.0.7 που αποτελούσε την πλέον πρόσφατη την χρονική περίοδο διεξαγωγής των πειραμάτων.

Από την διεξαγωγή των πειραμάτων προέκυψαν αρκετά χρήσιμα συμπεράσματα. Ορισμένα από αυτά τα συμπεράσματα ήταν αναμενόμενα (αφού ήταν καταγεγραμμένα) και απλά επιβεβαιώθηκαν πειραματικά, ενώ άλλα αποτελούν γνώση που δεν ήταν καταγεγραμμένη στο documentation των συστημάτων αρχείων και σε papers που τα περιέγραφαν.
Στη συνέχεια περιγράφουμε κάθε ένα από τα πειράματα και τα συμπεράσματα που προέκυψαν από αυτό, σε ξεχωριστή υποενότητα.

8.3.1 Πείραμα 1

Σκοπός

Σκοπός αυτού του πειράματος είναι να δούμε αν τα δύο υπό μελέτη συστήματα

αρχείων χρησιμοποιούν Cooperative Caching.
Το Cooperative Caching είναι μια τεχνική που στοχεύει στην βελτίωση των επιδόσεων των δικτυακών συστημάτων αρχείων (η γενική κατηγορία στην οποία εντάσσονται τα GFS και GPFS) μέσω της συλλογικής χρήσης των cache όλων των κόμβων που χρησιμοποιούν το σύστημα αρχείων. Με αυτό τον τρόπο όταν ένας κόμβος κάνει μια αίτηση προσπέλασης σε δεδομένα του αποθηκευτικού μέσου, ελέγχεται η cache όλων των κόμβων για την ύπαρξη των δεδομένων και όχι μόνο η δική του.

Τρόπος Διεξαγωγής

Για να δούμε αν τα συστήματα αρχείων GFS και GPFS κάνουν χρήση του Cooperative Caching, κάνουμε το εξής πείραμα: Αρχικά ένας κόμβος Α διαβάζει ένα ολόκληρο αρχείο με χρήση του mBench ενώ παράλληλα καταγράφονται τα blocks δεδομένων του δίσκου που προσπελάζονται, με την χρήση του blktrace πάνω στον δίσκο στον οποίο έχει στηθεί το σύστημα αρχείων. Εναλλακτικά στην περίπτωση του GPFS, μπορούμε να χρησιμοποιήσουμε το GPFS Trace ρυθμισμένο ώστε να καταγράφονται μόνο δεδομένα που αφορούν την ανάγνωση blocks δεδομένων από τον δίσκο. Και στις δύο περιπτώσεις η καταγραφή μας πληροφορεί για τα blocks δεδομένων που τελικά αναγνώσθηκαν από τον δίσκο από τον κόμβο Α.
Το αρχείο που διαβάζει ο κόμβος Α πρέπει να είναι αρκετά μικρό ώστε να χωρά εξ ολοκλήρου στην cache. Επιβεβαιώνουμε το γεγονός ότι το αρχείο βρίσκεται ολόκληρο στην cache με την καταγραφή των blocks δεδομένων του δίσκου που προσπελαύνονται σε μια δεύτερη ανάγνωση του ίδιου αρχείου από τον κόμβο Α. Η δεύτερη αυτή καταγραφή βλέπουμε πως δεν περιλαμβάνει κανένα block του δίσκου, συνεπώς όλα τα δεδομένα του αρχείου αναγνώστηκαν από την cache.
Στη συνέχεια οι κόμβοι Β και Γ διαβάζουν το ίδιο αρχείο με τον ίδιο τρόπο ενώ ταυτόχρονα γίνονται καταγραφές. Σημειώνουμε ότι στους κόμβους Β και Γ η ανάγνωση του αρχείου είναι η πρώτη ενέργεια που κάνουν, πράγμα που εξασφαλίζει ότι το αρχείο σίγουρα δεν βρίσκεται στην δική τους cache.

Συμπέρασμα

Συγκρίνοντας τις καταγραφές που πήραμε από τους κόμβους Β και Γ με την αρχική καταγραφή από τον κόμβο Α, μπορούμε να δούμε αν διαβάστηκαν δεδομένα από τον δίσκο ή τα δεδομένα αυτά εστάλησαν από την cache του κόμβου Α (και του Β αν ο Γ είναι ο τελευταίος που διαβάζει το αρχείο) . Όπως είπαμε το πείραμα έγινε με τέτοιο τρόπο ώστε να είμαστε βέβαιοι πως το αρχείο δεν βρίσκεται στην cache των Β και Γ πριν την ανάγνωσή του.
Η σύγκριση των καταγραφών του κόμβου Β και του Γ με την καταγραφή του Α κάνοντας χρήση ενός από τα προγράμματα comparer, δείχνει ότι όλοι οι κόμβοι διαβάζουν τα ίδια blocks του δίσκου (με διαφορά ίσως ελάχιστων blocks), συνεπώς δεν χρησιμοποιείται ούτε η δικιά τους cache (πράγμα που το ξέραμε) ούτε η cache των άλλων κόμβων. Το γεγονός αυτό ισχύει και για τα δύο συστήματα αρχείων και αποδεικνύει πως δεν γίνεται χρήση του cooperative caching.

Πρακτικό Αντίκτυπο Ανυπαρξίας Cooperative Caching

Η ανυπαρξία του Cooperative Caching έχει ως κύριο μειονέκτημα το γεγονός ότι απαιτούνται πιο συχνά από τους κόμβους, προσβάσεις στο κοινό αποθηκευτικό μέσο. Αυτό διότι στην περίπτωση που δεν υπάρχουν τα απαραίτητα δεδομένα στην δική τους cache, πρέπει να γίνει πρόσβαση στον δίσκο.
Τα πλεονεκτήματα που συνεπάγεται η ανυπαρξία του Cooperative Caching στο GFS και στο GPFS είναι δύο: Πρώτον, η διατήρηση της συνέπειας των cache των κόμβων είναι πιο απλή καθώς δεν απαιτείται η ύπαρξη του πολύπλοκου πρωτοκόλλου συντονισμού τους. Όπως έχουμε ξαναπεί, στην περίπτωση των δύο συστημάτων αρχείων που μελετάμε, η συνέπεια της cache κάθε κόμβου είναι διατηρείται χάρη στον συντονισμό των ενεργειών της με την διαχείριση των κλειδωμάτων. Δεύτερον, δεν φορτίζεται το δίκτυο διασύνδεσης με το επιπλέον φορτίο του πρωτοκόλλου συντονισμού των cache και της μεταφοράς δεδομένων από την cache ενός κόμβου στην μνήμη άλλων κόμβων.

8.3.2 Πείραμα 2

Σκοπός

Ο σκοπός αυτού του πειράματος είναι να δούμε στην πράξη την διαφορά που έχουμε αναφέρει στην ενότητα 4.8, ότι δηλαδή η συμπεριφορά των GPFS και GFS διαφέρει στην περίπτωση που έχουν στις cache τους ένα αρχείο που στη συνέχεια τροποποιείται από άλλο κόμβο.

Τρόπος Διεξαγωγής

Ο τρόπος με τον οποίο διεξάγεται το πείραμα είναι ο εξής: Αρχικά ένας κόμβος Α διαβάζει ένα αρχείο. Η ανάγνωση περιλαμβάνει όλα τα δεδομένα του αρχείου και είναι απλή σειριακή με την χρήση του προγράμματος mBench. Παράλληλα με την ανάγνωση αυτή γίνεται καταγραφή με την χρήση του blktrace και εναλλακτικά του GPFS Trace. Πρέπει να δοθεί προσοχή ώστε το αρχείο που θα αναγνωστεί από τον κόμβο Α να χωράει εξ ολοκλήρου στην cache. Για την εξασφάλιση του γεγονότος αυτού, χρησιμοποιήθηκε ένα αρχείο μεγέθους 50MB. Το γεγονός ότι το αρχείο βρίσκεται όλο στην cache του κόμβου Α μετά την πρώτη ανάγνωση, επιβεβαιώνεται με μια νέα καταγραφή των blocks δεδομένων που ζητήθηκαν από τον δίσκο όταν γίνεται μια νέα σειριακή ανάγνωση του αρχείου από τον κόμβο Α. Στην καταγραφή αυτή βλέπουμε ότι δεν ζητήθηκαν καθόλου δεδομένα από τον δίσκο, συνεπώς όλα τα δεδομένα του αρχείου υπήρχαν στην cache.
Στη συνέχεια του πειράματος ένας κόμβος Β κάνει χρήση του προγράμματος mWrite για να αλλάξει μια μικρή περιοχή του αρχείου. Αφού ο Β κάνει αυτή την τροποποίηση, ο Α ξαναδιαβάζει το αρχείο με τον ίδιο τρόπο, ενώ παράλληλα γίνεται καταγραφή και αυτής ανάγνωσης.
Για την πληρότητα του πειράματος, διεξάχθηκε τρεις φορές για κάθε σύστημα αρχείων, με την διαφορά να είναι το σημείο του αρχείου που τροποποιήθηκε κάθε φορά.

Συμπέρασμα

Στην περίπτωση του GPFS η νέα καταγραφή μας δείχνει ότι διαβάστηκε μόνο το block του αρχείου που τροποποιήθηκε. Αντίθετα η σύγκριση των καταγραφών της πρώτης ανάγνωσης και της ανάγνωσης μετά την τροποποίηση για το GFS με χρήση του προγράμματος comparer, δείχνει ότι διαβάστηκαν ακριβώς τα ίδια δεδομένα.
Συνεπώς επιβεβαιώνεται αυτό που είχε ειπωθεί στη θεωρία, ότι δηλαδή όταν χρησιμοποιείται το GPFS δεν χρειάζεται να διαβαστεί εκ νέου παρά μόνο το τροποποιημένο κομμάτι του αρχείου, σε αντίθεση με το GFS που σε κάθε περίπτωση διαβάζεται όλο το αρχείο από την αρχή.

Τεχνικό αίτιο ύπαρξης της διαφοράς

Όπως έχουμε εξηγήσει και στην ενότητα 4.8, ο λόγος που διαφέρει το GPFS από το GFS στο ότι μπορεί να διαβάσει μόνο τα blocks που τροποποιήθηκαν, είναι ότι πρώτο εκχωρεί κλειδώματα δυναμικά για μέρη του αρχείου, ενώ το GFS εκχωρεί κλειδώματα μόνο για ολόκληρο το αρχείο.
Όταν δηλαδή ένας κόμβος θέλει να γράψει σε ένα αρχείο, τότε στο GPFS ανακαλούνται τα κλειδώματα των άλλων κόμβων που αντιστοιχούν μόνο στα προς τροποποίηση δεδομένα, με αποτέλεσμα την ακύρωση (και ενδεχομένως εγγραφή στον αποθηκευτικό χώρο αν έχουν υποστεί τροποποίηση) μόνο αυτών των δεδομένων από τις cache. Στο GFS όταν ένας κόμβος θέλει να γράψει σε ένα αρχείο, ανακαλείται το συνολικό κλείδωμα του αρχείου από τους άλλους κόμβους, με αποτέλεσμα την ακύρωση όλων των δεδομένων των cache τους που σχετίζονται με το αρχείο.

Πρακτικό αντίκτυπο της συγκεκριμένης διαφοράς των δύο συστημάτων αρχείων

Το αποτέλεσμα της διαφοράς που περιγράψαμε παραπάνω, είναι ότι το GPFS είναι κατάλληλο για διαμοιρασμό του ίδιου αρχείου σε πολλούς κόμβους, σε αντίθεση με το GFS, όπου ναι μεν η δυνατότητα αυτή υπάρχει, αλλά έχει ως αποτέλεσμα τις άσχημες I/O επιδόσεις.

Τροποποίηση πειράματος

Μια τροποποίηση που μπορεί να έγινε σε αυτό το πείραμα για να διευκρινιστεί περαιτέρω η συμπεριφορά των cache των GPFS και GFS είναι να μην γίνεται η τροποποίηση του αρχείου από τον κόμβο B, αλλά από τον κόμβο A. Δηλαδή ο ίδιος κόμβος που διάβασε αρχικά το αρχείο, θα είναι και αυτός που θα το τροποποιήσει.
Ο σκοπός του πειράματος αυτού είναι να δούμε από πού θα διαβαστούν σε αυτή την περίπτωση τα δεδομένα.
Η εξέταση των καταγραφών που προέκυψαν από την ανάγνωση που επακολούθησε της τροποποίησης, δείχνει ότι και στα δύο συστήματα αρχείων, η ανάγνωση έγινε από την cache και όχι από τον δίσκο.

8.3.3 Πείραμα 3

Σκοπός

Ο σκοπός του συγκεκριμένου πειράματος είναι να διαπιστώσουμε κατά πόσο τα δύο συστήματα αρχείων χρησιμοποιούν prefetching σε σενάρια σειριακής και αντίστροφης σειριακής ανάγνωσης. Το prefetching είναι η διαδικασία κατά την οποία το σύστημα αρχείων κάνει ανάγνωση περισσοτέρων δεδομένων από όσα ζητήθηκαν από μια εφαρμογή. Τα επιπλέον δεδομένα μπαίνουν στην cache που διατηρεί το σύστημα αρχείων, προκειμένου να αξιοποιηθούν σε πιθανή μελλοντική ανάγνωση που θα πραγματοποιήσει η εφαρμογή. Ουσιαστικά το prefetching εκμεταλλεύεται το γεγονός ότι πολλές εφαρμογές κάνουν σειριακή προσπέλαση των αρχείων, ώστε να επιτυγχάνονται υψηλότερες I/O επιδόσεις.
Για το GPFS γνωρίζουμε από το documentation του ότι κάνει prefetching, αλλά δεν έχουμε γνώση για την έκτασή του.
Όσον αφορά το GFS δεν έχουμε από τη θεωρία πληροφορίες που να αφορούν το prefetching.

Τρόπος διεξαγωγής

Αρχικά θέλουμε να δούμε κατά πόσον γίνεται prefetching σε περιπτώσεις σειριακής ανάγνωσης των αρχείων. Χρησιμοποιούμε τρία αρχεία μεγέθους 50, 75 και 100MB αντίστοιχα και κάνουμε τα εξής βήματα για το καθένα ξεχωριστά:
Διαβάζουμε με την χρήση του mBench το πρώτο μισό του αρχείο και καταγράφουμε με την χρήση του blktrace τα δεδομένα του δίσκου που προσπελάστηκαν και αντιστοιχούν στην ανάγνωση του πρώτου μισού του αρχείου. Σημειώνουμε επίσης το συνολικό ποσό των δεδομένων που ζητήθηκε από τον δίσκο, όπως αυτό εμφανίζεται στο τέλος της καταγραφής που μας δίνει το blktrace. Στη συνέχεια επαναλαμβάνουμε την ίδια διαδικασία για το δεύτερο μισό του αρχείου, χωρίς να έχει παρεμβληθεί κάποια άλλη ενέργεια. Όπως θα πούμε παρακάτω και στο συμπέρασμα του πειράματος, η σύγκριση των δύο καταγεγραμμένων ποσοτήτων δεδομένων μας πληροφορεί για την ύπαρξη ή όχι prefetching.
Το επόμενό μας βήμα είναι να επαναλάβουμε το πείραμα αυτή τη φορά όμως διαβάζοντας κάθε φορά από την αρχή μέχρι το τέλος του, κάθε τέταρτο του αρχείου χρησιμοποιώντας την αντίστοιχη δυνατότητα του mBench. Επίσης σημειώνουμε το ποσό των δεδομένων που ζητήθηκαν από τον δίσκο σε κάθε περίπτωση.
Αφού λοιπόν γίνουν όλα τα παραπάνω βήματα για κάθε αρχείο ξεχωριστά, σειρά παίρνει η πειραματική διαδικασία για την εξαγωγή αποτελεσμάτων σχετικά με το prefetching της αντίστροφη σειριακή ανάγνωση. Το πείραμα αυτό διεξάγεται με τον ίδιο τρόπο με το προηγούμενο, με την μόνη διαφορά ότι η σειρά ανάγνωσης των μισών και των τετάρτων των αρχείων είναι ανεστραμμένη.

Συμπεράσματα

Αφού λοιπόν γίνουν όλα αυτές οι αναγνώσεις για κάθε ένα σύστημα αρχείων ξεχωριστά, συγκεντρώνονται στους παρακάτω πίνακες τα ποσά δεδομένων που αναγνώστηκαν σε κάθε περίπτωση από τον δίσκο.

Αρχείο 50MB

	Σειριακή Προσπέλαση		**Αντίστροφη Σειριακή Προσπέλαση**	
Μέρος του αρχείου	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GPFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GPFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GFS (MB)**
1ο Μισό	28.5	25.25	28.25	25
2ο Μισό	21.5	24.75	21.75	25
1ο Τέταρτο	16.25	12.75	15.75	12.5
2ο Τέταρτο	12.25	12.5	12.5	12.5
3ο Τέταρτο	12.5	12.5	12.5	12.5
4ο Τέταρτο	9	12.25	9.25	12.5

Πίνακας 8.1: Πείραμα 3, Αρχείο 50MB

Αρχείο 75MB

	Σειριακή Προσπέλαση		**Αντίστροφη Σειριακή Προσπέλαση**	
Μέρος του αρχείου	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GPFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GPFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GFS (MB)**
1ο Μισό	41	37.75	41	37.5
2ο Μισό	34	37.25	34	37.5
1ο Τέταρτο	22.25	19	22	18.75
2ο Τέταρτο	18.75	18.75	18.75	18.75
3ο Τέταρτο	18.75	18.75	18.75	18.75
4ο Τέταρτο	15.25	18.5	15.5	18.75

Πίνακας 8.2: Πείραμα 3, Αρχείο 75MB

Αρχείο 100MB

	Σειριακή Προσπέλαση		Αντίστροφη Σειριακή Προσπέλαση	
Μέρος του αρχείου	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GPFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GPFS (MB)**	**Ποσό αναγνωσθέντων από τον δίσκο δεδομένων στο GFS (MB)**
1ο Μισό	53.75	50.25	53.25	50
2ο Μισό	46.25	49.75	46.75	50
1ο Τέταρτο	28.5	25.25	28.5	25
2ο Τέταρτο	25	25	24.75	25
3ο Τέταρτο	25	25	25	25
4ο Τέταρτο	21.5	24.75	21.75	25

Πίνακας 8.3: Πείραμα 3, Αρχείο 100MB

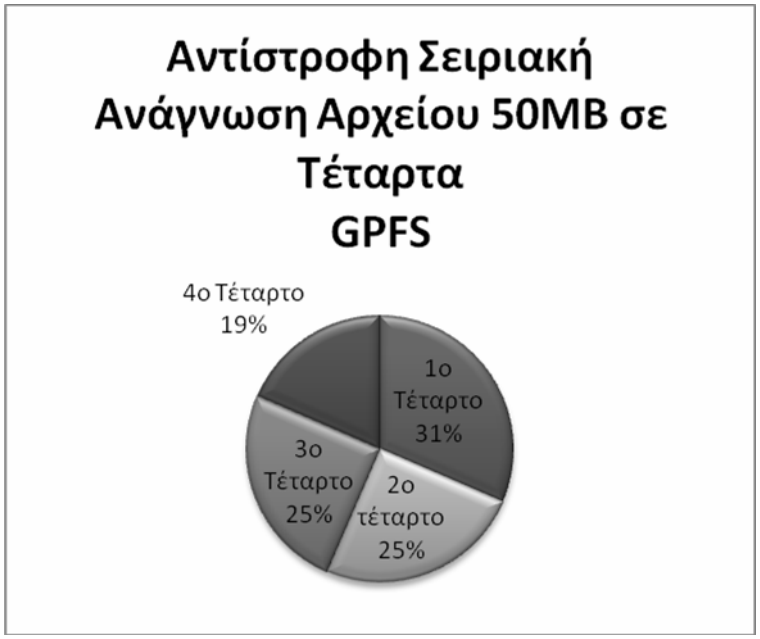
Για την περίπτωση των αναγνώσεων στο αρχείο των 50MB δίνουμε σε μορφή διαγράμματος το ποσοστό των δεδομένων που αναγνώστηκαν, επί του συνολικού μεγέθους του αρχείου. Τα αντίστοιχα διαγράμματα για τα άλλα αρχεία μας δίνουν την δυνατότητα να βγάλουμε τα ίδια ακριβώς συμπεράσματα και δεν κρίνεται σκόπιμο να παρατεθούν.

Σχήμα 8.1: Σειριακή Ανάγνωση Αρχείου σε Τέταρτα - GPFS

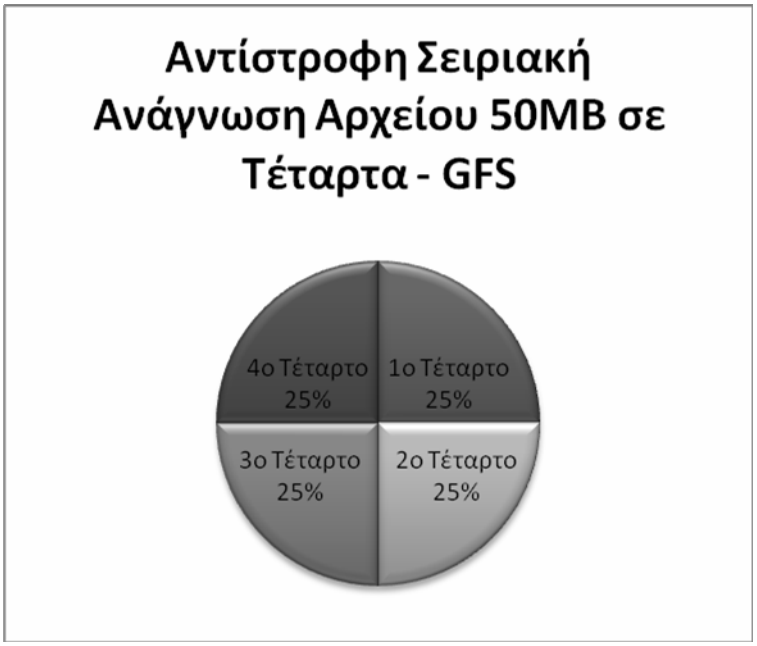
Σχήμα 8.2: Σειριακή Ανάγνωση Αρχείου σε Τέταρτα - GFS

Σχήμα 8.3: Σειριακή Ανάγνωση Αρχείου σε Μισά - GPFS

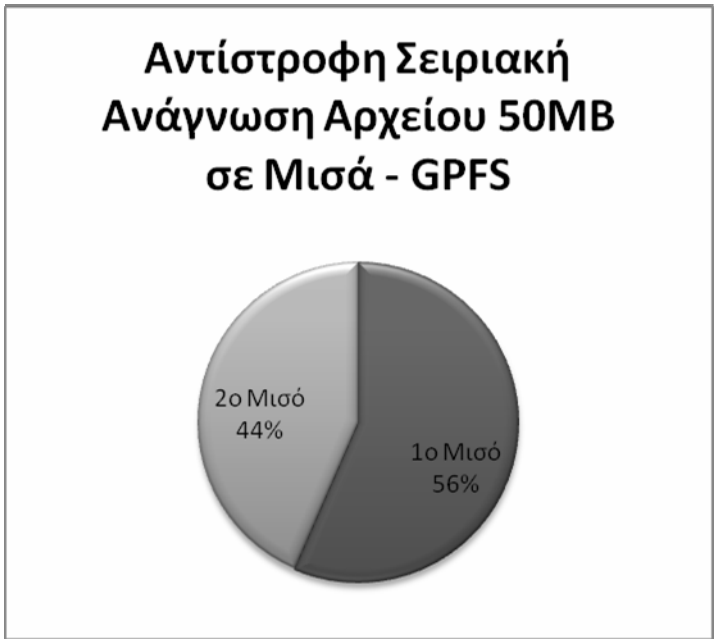
Σχήμα 8.4: Σειριακή Ανάγνωση Αρχείου σε Μισά - GFS



Σχήμα 8.5: Αντίστροφη Σειριακή Ανάγνωση Αρχείου σε Τέταρτα - GPFS



Σχήμα 8.6: Αντίστροφη Σειριακή Ανάγνωση Αρχείου σε Τέταρτα - GFS



Σχήμα 8.7: Αντίστροφη Σειριακή Ανάγνωση Αρχείου σε Μισά - GPFS

Σχήμα 8.8: Αντίστροφη Σειριακή Ανάγνωση Αρχείου σε Τέταρτα - GFS

Από τους παραπάνω πίνακες και τις εικόνες μπορούμε να βγάλουμε τα εξής συμπεράσματα:
Το GFS κάνει ελάχιστο prefetching και μόνο στην περίπτωση της σειριακής ανάγνωσης. Στην περίπτωση της αντίστροφης σειριακής ανάγνωσης δεν γίνεται καθόλου prefetching.
Αντίθετα το GPFS κάνει εκτενή χρήση του prefetching. Το γεγονός αυτό το ξέραμε καθώς αναφέρεται στο documentation του, αλλά δεν γνωρίζαμε το εύρος του.

Παρακάτω εξηγούνται ποιοτικά πως προκύπτουν τα νούμερα του prefetching για το GPFS για την ανάγνωση των τετάρτων των αρχείων.
Από τα νούμερα βλέπουμε πως έχουν γίνει περισσότερες προσβάσεις στον δίσκο από τις απαιτούμενες για την ανάγνωση του πρώτου τετάρτου του αρχείου. Οι απαιτούμενες αναγνώσεις του mBench κανονικά θα δημιουργούσαν κίνηση προς τον δίσκο ίση με ακριβώς το ένα τέταρτο του μεγέθους. Άρα βλέποντας το μέγεθος της κίνησης προς τον δίσκο, διαπιστώνουμε πως έχουν αναγνωριστεί περισσότερα από τα απαιτούμενα byte. Οι επόμενες δύο αναγνώσεις (δηλαδή οι αναγνώσεις του 2ου και του 3ου τετάρτου αντίστοιχα) φέρνουν ακριβώς τόσα δεδομένα όσα αντιστοιχούν σε ένα τέταρτο του αρχείου. Αυτό σημαίνει ότι και πάλι έχει γίνει ανάγνωση από το δίσκο περισσοτέρων δεδομένων από όσα απαιτούσε η ανάγνωση του αντίστοιχου τετάρτου, αφού η αφετηρία της ανάγνωσης από τον δίσκο δεν ήταν τα δεδομένα που αντιστοιχούν στην αρχή του συγκεκριμένου τετάρτου, αλλά λίγο πιο κάτω αφού η προηγούμενη ανάγνωση έκανε prefetching και άρα τα αρχικά δεδομένα του τετάρτου διαβάστηκαν από την cache. Η ανάγνωση του τελευταίου τετάρτου του αρχείου έχει ως αποτέλεσμα να διαβάζονται από τον δίσκο αρκετά λιγότερα δεδομένα από ότι θα απαιτούσε κανονικά η ανάγνωση, αφού τα υπόλοιπα (δηλαδή δεδομένα που αντιστοιχούν στην αρχή του τελευταίου τετάρτου) είναι στην cache λόγω του prefetching που έγινε κατά την ανάγνωση του προηγούμενου τετάρτου.
Τα ίδια πράγματα διαπιστώνουμε όταν διαβάζουμε αρχικά το πρώτο μισό ενός αρχείου και στη συνέχεια το δεύτερο μισό του ίδιου αρχείου.

Πρακτικό αντίκτυπο του prefetching

Το prefetching μπορεί να αυξήσει σημαντικά τις I/O επιδόσεις εφαρμογών που χρησιμοποιούν σειριακή ή αντίστροφη σειριακή προσπέλαση στα αρχεία. Αυτό διότι κάθε απόπειρα ανάγνωσης (εκτός από την πρώτη) ολοκληρώνεται πιο γρήγορα εφόσον κάποια από τα απαιτούμενα δεδομένα βρίσκονται στην cache και δεν χρειάζεται να διαβαστούν από το αποθηκευτικό μέσο.
Όταν χρησιμοποιείται το GPFS πάνω από ένα πραγματικό SAN το πλεονέκτημα της ταχύτητας που επιφέρει η χρήση του prefetching είναι ακόμα μεγαλύτερο αφού τα δεδομένα των αρχείων μοιράζονται στους δίσκους μέσω του striping και άρα το prefetching μπορεί να γίνει ακόμα πιο γρήγορα αφού μπορεί να εκμεταλλευτεί τον ρυθμό διαμεταγωγής όλων των διαθέσιμων δίσκων.

8.3.4 Πείραμα 4

Σκοπός

Ο σκοπός αυτού του πειράματος είναι να διαπιστώσουμε ποια πολιτική εξόδου από την cache χρησιμοποιεί κάθε ένα από τα δύο υπό μελέτη συστήματα αρχείων.
Η πληροφορία αυτή δεν είναι καταγεγραμμένη στο documentation των δύο συστημάτων αρχείων.

Τρόπος Διεξαγωγής

Τρεις είναι οι στρατηγικές που χρησιμοποιούνται γενικά σε caching συστήματα. Αυτές είναι οι MRU (Most Recently Used), LRU (Least Recently Used) και FIFO (First In First Out).
Η πρώτη στρατηγική δεν έχει νόημα να χρησιμοποιηθεί σε I/O συστήματα καθώς αγνοεί τελείως την σειριακή προσπέλαση των δεδομένων που χρησιμοποιούν πολλές εφαρμογές. Διαπιστώνουμε πως η στρατηγική αυτή όντως δεν ισχύει με το παρακάτω πείραμα: Διαβάζουμε ένα μεγάλο αρχείο, το οποίο γνωρίζουμε ότι είναι λίγο μεγαλύτερο από το μέγεθος της cache (Στην περίπτωση του GPFS το μέγεθος της cache αφέθηκε στην default τιμή των 60MB ενώ στην περίπτωση του GFS όταν τα εικονικά μηχανήματα είχαν μνήμη 256MB και δεν έτρεχαν επιπλέον λογισμικό, βρέθηκε πειραματικά ότι το μέγεθός της ήταν γύρω στα 110MB). Ταυτόχρονα καταγράφουμε τα blocks των δεδομένων που ζητήθηκαν είτε με τη χρήση του blktrace, είτε με την χρήση του GPFS Trace (για την περίπτωση του GPFS). Στη συνέχεια επαναλαμβάνουμε την ανάγνωση του ίδιου αρχείου και ταυτόχρονα κάνουμε την ίδια καταγραφή. Όταν συγκρίνουμε τις δύο καταγραφές με την χρήση ενός από τα προγράμματα comparer βλέπουμε ότι οι δύο καταγραφές περιλαμβάνουν ακριβώς τα ίδια blocks. Αυτό σημαίνει ότι και στην δεύτερη ανάγνωση κανένα block δεδομένων δεν διαβάστηκε από την cache. Αν η στρατηγική εξόδου ήταν MRU, αυτό θα σήμαινε ότι μετά την πρώτη ανάγνωση θα εξέρχονταν από την cache τα τελευταία δεδομένα του αρχείου, αφήνοντας τα πρώτα στην cache και άρα θα έπρεπε να αναγνωστούν από εκεί. Στην υποενότητα των συμπερασμάτων εξηγούνται τα πειραματικά αυτά αποτελέσματα στην περίπτωση που χρησιμοποιείται στρατηγική εξόδου FIFO ή LRU.

Εφόσον διαπιστώσαμε ότι δεν χρησιμοποιείται MRU, πρέπει να κάνουμε ένα πείραμα που θα διασαφηνίσει αν χρησιμοποιείται LRU ή FIFO πολιτική εξόδου.
Το πείραμα που κάνουμε είναι το εξής: Χρησιμοποιούμε τρία αρχεία ίδιου μεγέθους, από τα οποία δύο μπορούν να χωρέσουν ταυτόχρονα στην cache. Έστω ότι τα αρχεία αυτά είναι τα Α, Β και Γ. Σε αυτό το πείραμα μόνο ένας κόμβος συμμετέχει στην ανάγνωση των αρχείων.
Αρχικά ο κόμβος διαβάζει το αρχείο Α και αμέσως μετά το Β κάνοντας χρήση της σειριακής ανάγνωσης του mBench. Ταυτόχρονα γίνεται καταγραφή των blocks του δίσκου που εμπλέκονται στην ανάγνωση. Η καταγραφή γίνεται με το blktrace ή εναλλακτικά με το GPFS Trace στην περίπτωση του GPFS. Βεβαιωνόμαστε ότι και τα δύο αρχεία βρίσκονται στην cache (διαβάζοντας τα πάλι και βλέποντας στις καταγραφές τους ότι δεν υπάρχει καμία πρόσβαση στον δίσκο). Εν συνεχεία διαβάζεται εκ νεόυ το πρώτο αρχείο, ώστε να είναι ταυτόχρονα το πρώτο που μπήκε στην cache αλλά και το τελευταίο που διαβάστηκε. Για να μάθουμε ποιο αρχείο θα βγει από την cache και συνεπώς να διαπιστώσουμε ποια πολιτική εξόδου εφαρμόζεται, ο κόμβος διαβάζει το τρίτο αρχείο. Το μέγεθος της cache αλλά και των αναγνωσθέντων αρχείων συνεπάγονται πως κάποια δεδομένα ενός από τα δύο πρώτα αρχεία έχουν εξέλθει από την cache καθώς δεν χωράνε (και σίγουρα δεν χρησιμοποιείται MRU).
Το τελικό βήμα είναι να διαβάσουμε το ένα από τα δύο αρχεία και να καταγράψουμε τα blocks του δίσκου που μεταφέρθηκαν. Το πείραμα εκτελέστηκε δύο φορές, με τη διαφορά ότι στο τελευταίο βήμα την πρώτη φορά διαβάστηκε το αρχείο Α ενώ την δεύτερη φορά διαβάστηκε το αρχείο Β.

Συμπεράσματα

Με τον τρόπο που έχει διεξαχθεί το πείραμα το αρχείο Α είναι το τελευταίο που χρησιμοποιήθηκε και το πρώτο που μπήκε. Για να είναι η πολιτική εξόδου FIFO, θα πρέπει να συμβούν δύο γεγονότα: 1^{ον}) Την πρώτη φορά που διεξάγεται το πείραμα και διαβάζεται στο τελευταίο βήμα το αρχείο Α, θα πρέπει η καταγραφή να δείξει ότι η ανάγνωση γίνεται από τον δίσκο. 2^{ον}) Την δεύτερη φορά που διεξάγεται το πείραμα και διαβάζεται στο τελευταίο βήμα το αρχείο Β, θα πρέπει η καταγραφή να δείξει ότι η ανάγνωση γίνεται από την cache.
Αντίστοιχα αν η πολιτική εξόδου από την cache είναι LRU θα πρέπει να συμβούν τα εξής: 1^{ον}) Την πρώτη φορά που διεξάγεται το πείραμα και διαβάζεται στο τελευταίο βήμα το αρχείο Α, θα πρέπει η καταγραφή να δείξει ότι η ανάγνωση γίνεται από την cache. 2^{ον}) Την δεύτερη φορά που διεξάγεται το πείραμα και διαβάζεται στο τελευταίο βήμα το αρχείο Β, θα πρέπει η καταγραφή να δείξει ότι η ανάγνωση γίνεται από την δίσκο.

Στην περίπτωση του GPFS διαπιστώνουμε ότι ισχύουν τα γεγονότα που αντιστοιχούν στην FIFO πολιτική εξόδου.
Δυστυχώς στην περίπτωση του GFS η κατάσταση είναι μπερδεμένη καθώς δεν συμβαίνουν τα γεγονότα που αντιστοιχούν σε κάποια από τις δύο πολιτικές εξόδου και συνεπώς δεν μπορέσαμε να βγάλουμε συμπέρασμα.

Στο σημείο αυτό εξηγούμε ποιοτικά αλλά και με την χρήση ενός παραδείγματος, γιατί κάθε ανάγνωση ενός αρχείου που δεν χωρά στην cache, καταλήγει σε ανάγνωση όλων των δεδομένων από τον δίσκο όταν χρησιμοποιείται στρατηγική FIFO ή LRU.

Έστω X το μέγεθος της cache και Y το μέγεθος του αρχείου όπου Y > X. Αρχικά διαβάζεται όλο το αρχείο σειριακά από μία εφαρμογή. Εξ' αιτίας της πολιτικής της cache και του γεγονότος ότι τα δεδομένα διαβάζονται σειριακά, θα μείνουν στην cache μόνο τα τελευταία X σε μέγεθος δεδομένα του αρχείου. Συνεπώς σε επόμενη σειριακή ανάγνωση του ίδιου αρχείου αν δεν έχει παρεμβληθεί κάποια άλλη ενέργεια, τα δεδομένα στην cache δεν θα είναι σε αντιστοιχία με αυτά που ζητά η εφαρμογή όταν ξεκινά να διαβάζει το αρχείο. Αυτό γιατί η εφαρμογή ξεκινά την ανάγνωση από την αρχή του αρχείου ενώ η cache περιέχει τα τελευταία X σε μέγεθος δεδομένα του. Αυτό συνεπάγεται ότι τα πρώτα Y –X δεδομένα διαβάζονται σίγουρα από τον δίσκο. Ίσως να περίμενε κανείς ότι όταν η εφαρμογή φτάσει να διαβάσει τα τελευταία X σε μέγεθος δεδομένα, να τα διάβαζε από την cache. Αυτό δεν πρόκειται να συμβεί καθώς όταν η εφαρμογή φτάσει να διαβάσει αυτά τα δεδομένα, η πολιτική εξόδου της cache θα έχει διώξει ήδη τα πρώτο Y – X κομμάτι των αρχικών X δεδομένων της (που αντιστοιχούσαν στα τελευταία X δεδομένα του αρχείου), αφού αυτά τα Y –X δεδομένα ήταν τα πρώτα που εισήχθησαν στην cache. Η διαδικασία αυτή συνεχίζεται με αποτέλεσμα κανένα μέρος των δεδομένων του αρχείου να μην διαβάζεται από την cache.
Αυτό φαίνεται καλύτερα στο επόμενο παράδειγμα. Έστω ότι διαβάζεται ένα αρχείο μεγέθους 75MB, ενώ το μέγεθος της cache είναι 60MB. Αρχικά η cache είναι άδεια.
Αυτό το δείχνουμε στο επόμενο σχήμα όπως η cache φαίνεται χωρισμένη σε τέσσερα νοητά τέταρτα και κάθε κομμάτι είναι κενό.

---	---	---	---

Αφού διαβαστεί την πρώτη φορά όλο το αρχείο σειριακά, στην cache μένουν τα τελευταία 60MB του αρχείου. Αυτό φαίνεται όπου σε κάθε τέταρτο της cache υπάρχουν τα αντίστοιχα τέσσερα τελευταία κομμάτια μεγέθους 15MB του αρχείου. Με πιο μεγάλη γραμματοσειρά φαίνονται τα δεδομένα που μπήκαν πρώτα στην cache (που επίσης είναι και αυτά που χρησιμοποιήθηκαν τελευταία).

15-30MB	30-45MB	45-60MB	60-75MB

Ξεκινάμε να δούμε τώρα την δεύτερη σειριακή ανάγνωση σε βήματα του ενός πέμπτου του αρχείου, δηλαδή σε βήματα των 15MB. Γνωρίζουμε ότι η cache περιέχει τα δεδομένα που φαίνονται στην παραπάνω εικόνα. Συνεπώς η ανάγνωση των πρώτων 15MB του αρχείου θα γίνει σίγουρα από τον δίσκο. Λόγω της πολιτική εξόδου που είναι FIFO ή LRU, τα δεδομένα που θα εξέλθουν από την cache είναι αυτά που περιέχονται στο πρώτο τέταρτο. Η εικόνα της cache λοιπόν μετά την ανάγνωση των πρώτων 15MB του αρχείου είναι η εξής:

0-15MB	***30-45MB***	45-60MB	60-75MB

Στο δεύτερο βήμα, πρέπει να αναγνωστεί το δεύτερο κομμάτι μεγέθους 15MB του αρχείου. Επειδή και πάλι δεν βρίσκεται στην cache, το κομμάτι αυτό θα διαβαστεί από τον δίσκο. Η εικόνα της cache μετά και από αυτή την ανάγνωση είναι η εξής:

0-15MB	15-30MB	***45-60MB***	60-75MB

Συνεχίζοντας με τον ίδιο τρόπο την ανάλυση βλέπουμε ότι και το τρίτο και το τέταρτο κομμάτι μεγέθους 15MB του αρχείου θα διαβαστεί από την cache. Μετά την ανάγνωση του τέταρτου κομματιού η cache θα έχει τα εξής δεδομένα:

0-15MB	15-30MB	30-45MB	45-60MB

Το μόνο κομμάτι του αρχείου που δεν έχει διαβαστεί είναι το κομμάτι που αντιστοιχεί στα τελευταία 15MB που όμως και πάλι θα πρέπει να γίνει από τον δίσκο, καθώς η cache δεν περιέχει τα δεδομένα αυτά. Αφού γίνει η ανάγνωσή τους, η εικόνα που παίρνουμε είναι η ίδια με την εικόνα παραθέσαμε κατά την έναρξη της ανάλυσης της δεύτερης ανάγνωσης του αρχείου. Συνεπώς κάθε ανάγνωση του ίδιου αρχείο θα έχει ως αποτέλεσμα την ανάγνωση όλων των δεδομένων από τον δίσκο.

Πρακτικό Αντίκτυπο Χρήσης FIFO Στρατηγικής

Η χρήση της πολιτικής FIFO από το GPFS έναντι της χρήσης της στρατηγικής LRU προξενεί μια σχετική έκπληξη. Αυτό διότι η LRU στρατηγική ταιριάζει καλύτερα στην σειριακή ανάγνωση των δεδομένων που κάνουν πολλές εφαρμογές.
Δύο είναι οι λόγοι για τους οποίους μάλλον επιλέχθηκε FIFO αντί για LRU στρατηγική εξόδου από την cache:

- Σε πολλές περιπτώσεις πρόσβασης στα αρχεία οι δύο στρατηγικές έχουν ως αποτέλεσμα την έξοδο ακριβώς των ίδιων δεδομένων από την cache.
- Η υλοποίηση της στρατηγικής FIFO είναι σαφώς πιο εύκολη και έχει ως αποτέλεσμα να γίνεται ταχύτερα η διαχείριση της cache.

Όπως είπαμε και στα συμπεράσματα η χρήση FIFO στρατηγικής έχει πολύ άσχημα αποτελέσματα στην περίπτωση ανάγνωσης όλων των δεδομένων αρχείων που είναι μεγαλύτερα από την cache. Όχι μόνο δεν αξιοποιείται η cache σε ενδεχόμενες επαναλαμβανόμενες αναγνώσεις του ίδιου μεγάλου αρχείου, αλλά επιπλέον ακυρώνονται τα πλεονεκτήματα που είχε το caching για όλα τα άλλα αρχεία που ενδεχομένως είχαν διαβαστεί σε προηγούμενες χρονικές στιγμές και είχαν μπει στην cache.
Σε αυτή την περίπτωση είναι ευθύνη του προγραμματιστή της εφαρμογής, να την δημιουργήσει με τέτοιο τρόπο ώστε να δίνει εντολή στο λειτουργικό σύστημα και κατ' επέκταση στο σύστημα αρχείων να μην γίνεται χρήση της cache όταν πρόκειται να διαβαστούν μεγάλες ποσότητες δεδομένων ενός αρχείου.

8.3.5 Πείραμα 5

Σκοπός

Σκοπός του πειράματος είναι συγκρίνουμε τα Full και Simple MPI I/O καθώς και να δούμε στην πράξη την χρήση του Data Shipping Mode του GPFS.

Τρόπος Διεξαγωγής

Η αρχική σκέψη για την διεξαγωγή του πειράματος αυτού ήταν να χρησιμοποιηθεί η εφαρμογή BTIO και να γίνει καταγραφή των προσβάσεων που γίνονται στον δίσκο όταν αυτή χρησιμοποιεί τρόπους ανάγνωσης Full MPI I/O, Simple MPI I/O καθώς και Full MPI I/O σε συνδυασμό με το Data Shipping(DS) Mode του GPFS. Ο πειραματισμός με τους πρώτους δύο τρόπου ανάγνωσης θα μπορούσε να γίνει και για τα δύο συστήματα αρχείων, ενώ ο τελευταίος μόνο για το GPFS.
Δυστυχώς δεν κατέστη δυνατό τεχνικά να ενεργοποιηθεί το Data Shipping Mode του GPFS για χρήση με το BTIO. Ο λόγος για το γεγονός αυτό ήταν ότι η βιβλιοθήκη MPICH δεν υποστηρίζει άμεσα το GPFS και συνεπώς δεν του περνά τα MPI Hints που εισήχθησαν στον κώδικα του BTIO και στόχευαν στην ενεργοποίηση του Data Shipping Mode. Η μόνη υλοποίηση της MPI βιβλιοθήκης που υποστηρίζει άμεσα το Data Shipping Mode είναι αυτή της IBM. Η απόκτηση της όμως δεν ήταν δυνατή καθώς δεν αποτελεί ελεύθερο λογισμικό.
Γι' αυτό το λόγο το BTIO χρησιμοποιήθηκε τελικά μόνο για να δούμε στην πράξη (μέσω των καταγραφών) την διαφορά που παρουσιάζουν το Full MPI I/O και το Simple MPI I/O. Θυμίζουμε από την περιγραφή του BTIO, ότι στο Full MPI I/O οι κόμβοι συνεργάζονται ώστε να γράψουν με αποδοτικό τρόπο τα αποτελέσματα της επεξεργασίας τους σε ένα μοναδικό αρχείο κάνοντας χρήση του I/O κομματιού της MPI βιβλιοθήκης, ενώ στο Simple MPI I/O ναι μεν οι κόμβοι γράφουν σε ένα αρχείο με χρήση του I/O κομματιού της MPI βιβλιοθήκης, αλλά δεν υπάρχει καμία συνεργασία μεταξύ τους για την εγγραφή αυτή. Ο κάθε κόμβος δηλαδή γράφει στο αρχείο τα αποτελέσματα της δικής του επεξεργασίας.
Για την μελέτη του Data Shipping Mode χρησιμοποιήθηκε τελικά το πρόγραμμα gpfsperf που έχει την δυνατότητα να ενεργοποιεί προγραμματιστικά το Data Shipping Mode.

Το πείραμα λοιπόν χωρίζεται σε δύο μέρη. Το πρώτο μέρος όπου μελετώνται οι διαφορές των Full MP I/O και Simple MPI I/O του BTIO και το δεύτερο όπου μελετάτε το Data Shipping Mode του GPFS με χρήση του gpfsperf. Περιγράφουμε ξεχωριστά κάθε μέρος.

Μέρος 1^{ο}

Σε αυτό μέρος εκτελούμε και στους τέσσερις κόμβους που χρησιμοποιούν τα συστήματα αρχείων, την εφαρμογή BTIO στην έκδοση για παράλληλη επεξεργασία από τέσσερις κόμβους. Η εκτέλεση για κάθε ένα από τα συστήματα αρχείων γίνεται τέσσερις φορές ώστε να καλυφθούν όλοι οι συνδυασμοί του μεγέθους του προβλήματος που επιλύεται (επιλέχθηκαν τα μεγέθη S και W του BTIO) και του τρόπου προσπέλασης (Full MPI I/O και Simple MPI I/O). Σε κάθε κόμβο γίνεται καταγραφή των προσβάσεων στον δίσκο με χρήση του blktrace.
Για να μπορέσουμε να βγάλουμε συμπεράσματα για την διαφορά των Full MPI I/O και Simple MPI I/O να συγκρίνουμε τα blocks που προσπελαύνει κάθε κόμβος και

δεν προσπελαύνουν οι άλλοι κόμβοι για τα δύο αυτά ήδη πρόσβασης.
Ένα παράδειγμα τέτοιας σύγκρισης είναι το παρακάτω:
Έστω ότι βλέπουμε από τις καταγραφές ότι κάθε κόμβος προσπελαύνει 100 blocks όταν χρησιμοποιείται η κλάση S του BTIO σε Full MPI I/O, ενώ σε Simple MPI I/O κάθε κόμβος προσπελαύνει 200 blocks.
Έστω επίσης ότι στο Full MPI I/O βρίσκουμε από την σύγκριση των καταγραφών ότι ο κόμβος 1 προσπελαύνει 80 blocks που δεν προσπελαύνει ο 2, 70 blocks που δεν προσπελαύνει ο 3 και 90 blocks που δεν προσπελαύνει ο 4. Τότε ο μέσος αριθμός block που προσπελαύνει ο κόμβος 1 και όχι οι άλλοι κόμβοι είναι 80 και το ποσοστό αυτών των block επί των συνολικών blocks που προσπέλασε ο κόμβος 1 είναι 80%. Το ποσοστό αυτό απλά αντικατοπτρίζει το γεγονός που βλέπουμε και από τα δεδομένα των blocks που προσπελαύνουν οι κόμβοι. Ότι δηλαδή οι προσβάσεις του κόμβου 1 γίνονται τις περισσότερες φορές σε δεδομένα που δεν θα προσπελαστούν από άλλους κόμβους.
Για την περίπτωση του Simple MPI I/O έστω ότι βρίσκουμε ότι το πλήθος blocks που προσπέλασε ο κόμβος 1 και δεν προσπέλασε ο κόμβος 2 είναι 100 ενώ τα αντίστοιχα νούμερα για τους κόμβους 3 και 4 είναι 60 και 50 blocks αντίστοιχα. Συνεπώς ο μεσος αριθμός των blocks που προσπελαύνει ο κόμβος 1 και δεν προσπελαύνουν οι άλλοι τρεις κόμβοι είναι 70, νούμερο που αντιστοιχεί στο 35% των συνολικών blocks που προσπέλασε ο κόμβος 1. Το μέσο ποσοστό δηλαδή των blocks που προσπελαύνει ο κόμβος 1 και δεν προσπελαύνουν οι άλλοι είναι 35% των blocks. Αυτό σημαίνει ότι αρκετά από τα blocks που προσπέλασε ο κόμβος 1, προσπελάστηκαν και από τους άλλους κόμβους.

Όπως καταλαβαίνει κανείς από το παραπάνω πείραμα, για να συγκρίνουμε τα δύο ήδη MPI I/O, θα πρέπει να βρούμε τα ποσά των δεδομένων που προσπελαύνει κάθε κόμβος και δεν προσπελαύνουν οι άλλοι. Για να είναι δυνατές οι συγκρίσεις, τα ποσά αυτά θα ανάγονται σε ποσοστά επί του συνολικού αριθμού blocks που διάβασε κάθε κόμβος.
Επίσης από το παράδειγμα βλέπουμε ότι για την εξαγωγή συμπερασμάτων είναι απαραίτητη η γνώση του αριθμού των blocks που προσπελάστηκαν από κάθε κόμβο καθώς και του πλήθους των αντίστοιχων blocks που δεν προσπελάστηκαν από τους άλλους κόμβους. Η γνώση αυτή προκύπτει από σύγκριση των καταγραφών μέσω ενός από τα προγράμματα comparer. Το πρόγραμμα αυτό διαβάζει δύο οποιοσδήποτε καταγραφές και στη συνέχεια τυπώνει το πλήθος των blocks που προσπελάστηκαν στις δύο καταγραφές καθώς και τον αριθμό των blocks που υπάρχουν στην μία καταγραφή και όχι την άλλη και αντίστροφα.

Μέρος 2^{ο}

Στο δεύτερο μέρος βλέπουμε πειραματικά την χρήση του Data Shipping Mode του GPFS. Χρησιμοποιούμε την εφαρμογή gpfsperf προκειμένου να γίνει παράλληλη σειριακή ανάγνωση ενός αρχείου από τρεις κόμβους. Το πείραμα εκτελείται δύο φορές, μία φορά χωρίς το Data Shipping Mode και μια δεύτερη με το τελευταίο ενεργοποιημένο.
Σε κάθε ένα κόμβο καταγράφουμε με το blktrace τα δεδομένα που ανέγνωσε κάθε κόμβος από τον δίσκο, ενώ με χρήση του GPFS Trace καταγράφουμε τα Data Shipping γεγονότα.
Για να δούμε αν το Data Shipping Mode όντως έχει διαφορά στην παράλληλη προσπέλαση των δεδομένων από τους κόμβους, αφού γίνει η καταγραφή, χρησιμοποιούμε ένα από τα προγράμματα comparer για να δούμε πόσα blocks δεδομένων

προσπελάζονται από πολλούς κόμβους στην περίπτωση που δεν είναι ενεργοποιημένο το Data Shipping Mode και στην περίπτωση που είναι.

Συμπεράσματα

Από τα δύο μέρη του πειράματος εξάγουμε διαφορετικά συμπεράσματα.

Μέρος 1ο
Όπως είπαμε στο πρώτο μέρος συγκρίνουμε το Full με το Simple MPI I/O του BTIO για τα δύο συστήματα αρχείων. Θεωρητικά αναμένουμε ότι η χρήση του Full MPI I/O θα πρέπει να έχει ως αποτέλεσμα οι κόμβοι να προσπελαύνουν λιγότερα κοινά δεδομένα από ότι με την χρήση του Simple MPI I/O.

Στη συνέχεια παρατίθενται πίνακες που περιλαμβάνουν το μέσο ποσοστό του πλήθους των blocks δεδομένων που προσπελάστηκαν από κάθε κόμβο αλλά όχι από τους άλλους σε κάθε εκτέλεση του πειράματος. Στους πίνακες αυτούς βλέπουμε πως τα ποσοστά δεν είναι ακριβώς τα ίδια για τα δύο συστήματα αρχείων. Το γεγονός αυτό συμβαίνει επειδή όπως έχουμε αναφέρει στο πείραμα 3 το GPFS κάνει prefetching με αποτέλεσμα να διαβάζονται ορισμένα παραπάνω blocks από αυτά που είναι απαραίτητα. Θα πρέπει επίσης να αναφέρουμε πως πρέπει να χρησιμοποιηθεί το ίδιο μέγεθος blocks δεδομένων στα δύο συστήματα αρχείων για να είναι τα ποσοστά στα δύο συστήματα αρχείων παρόμοια. Σε αυτή την περίπτωση τα αποτελέσματα είναι παρόμοια καθώς το πλήθος των block που διαβάζεται καθορίζεται από τοBTIO και όχι το σύστημα αρχείων (αν εξαιρέσουμε το prefetching).

GFS – Μικρότερο δυνατό μέγεθος προβλήματος (κλάση S)

	Μέσο ποσοστό των blocks που προσπελαύνει ένας κόμβου και δεν προσπελάζονται από τους άλλους κόμβους, επί των συνολικών blocks που προσπελαύνει ο κόμβος	
Κόμβοι	**Full MPI I/O**	**Simple MPI I/O**
Κόμβος 1	74.1	40.2
Κόμβος 2	63.5	17.4
Κόμβος 3	73.4	18.0
Κόμβος 4	78.2	22.4

Πίνακας 8.4: Πείραμα 5, GFS, κλάση S

GFS – Μικρό μέγεθος προβλήματος (κλάση W)

	Μέσο ποσοστό των blocks που προσπελαύνει ένας κόμβου και δεν προσπελάζονται από τους άλλους κόμβους, επί των συνολικών blocks που προσπελαύνει ο κόμβος	
Κόμβοι	**Full MPI I/O**	**Simple MPI I/O**
Κόμβος 1	69.8	17.6
Κόμβος 2	65.0	13.7
Κόμβος 3	74.3	7.6

Κόμβος 4	66.0	7.7

Πίνακας 8.5: Πείραμα 5, GFS, κλάση W

GPFS – Μικρότερο δυνατό μέγεθος προβλήματος (κλάση S)

	Μέσο ποσοστό των blocks που προσπελαύνει ένας κόμβου και δεν προσπελάζονται από τους άλλους κόμβους, επί των συνολικών blocks που προσπελαύνει ο κόμβος	
Κόμβοι	**Full MPI I/O**	**Simple MPI I/O**
Κόμβος 1	74.2	40.3
Κόμβος 2	63.5	17.4
Κόμβος 3	73.5	18.1
Κόμβος 4	78.3	22.5

Πίνακας 8.6: Πείραμα 5, GPFS, κλάση S

GPFS – Μικρό μέγεθος προβλήματος (κλάση W)

	Full MPI I/O	**Simple MPI I/O**
	69.9	17.7
	65.1	13.8
	74.4	7.7
	66.0	7.7

Πίνακας 8.7: Πείραμα 5, GPFS, κλάση W

Τα αποτελέσματα που βλέπουμε στον παραπάνω πίνακα επιβεβαιώνουν εν πολλοίς τα θεωρητικά αναμενόμενα.
Βλέπουμε δηλαδή πως η χρήση του Full MPI I/O, έχει ως αποτέλεσμα οι κόμβοι να διαβάζουν λιγότερα κοινά δεδομένα. Αυτό φαίνεται αφού το μέγεθος που δείχνει το ποσοστό των διαφορετικών blocks που προσπελάζονται, αυξάνεται πολύ όταν χρησιμοποιείται το Full MPI I/O αντί του Simple MPI I/O.
Από τους πίνακες βλέπουμε επίσης πως το σύστημα αρχείων που χρησιμοποιείται παίζει πολύ μεγάλο ρόλο στην διαφοροποίηση των Full και Simple MPI I/O. Συγκεκριμένα η χρήση του GPFS αναδεικνύει πολύ περισσότερο το Full MPI I/O από ότι η χρήση του GFS.

Αν και όπως εξηγήθηκε στην ενότητα 6.4 δεν είναι δυνατή η μέτρηση χρόνων εκτέλεσης, (εφόσον χρησιμοποιούμε Virtual Hardware με cpu sharing και παρεμβάλονται πολλά επίπεδα λογισμικού κατά την πραγματοποίηση I/O ενεργειών), εντούτοις παρατηρήθηκαν πολύ μεγάλες διαφορές στους χρόνους εκτέλεσης του BTIO. Συγκριμένα η εκτέλεση με χρήση του Full MPI I/O ήταν αρκετά ταχύτερη από

την εκτέλεση με χρήση του Simple MPI I/O. Επίσης οι χρόνος εκτέλεσης των πειραμάτων όταν γίνονταν χρήση του GPFS ήταν πολύ μικρότεροι από τον χρόνους εκτέλεσης των αντίστοιχων πειραμάτων όταν γίνονταν χρήση του GFS.
Ο λόγος που οι χρόνοι εκτέλεσης με χρήση του Full MPI I/O είναι μικρότεροι από του αντίστοιχους με χρήση του Simple MPI I/O είναι ο μικρότερος αριθμός συγκρούσεων για κλειδώματα. Πιο αναλυτικά, εφόσον στην περίπτωση του Full MPI I/O οι κόμβοι προσπελαύνουν μικρότερο αριθμό κοινών δεδομένων από ότι στην περίπτωση του Simple MPI I/O, δημιουργούνται λιγότερες συγκρούσεις για κλειδώματα, συνεπώς είναι λιγότερες οι φορές που οι κόμβοι αναγκάζονται να περιμένουν την απόκτηση ενός κλειδώματος για να μπορέσουν να ξεκινήσουν την ανάγνωση ή την εγγραφή που τους αντιστοιχεί.
Ο κύριος λόγος που η εκτέλεση του BTIO είναι ταχύτερη όταν χρησιμοποιείται GPFS αντί του GFS ως υποκείμενο σύστημα αρχείων, είναι το granularity του ελέγχου κλειδωμάτων. Όπως έχουμε πει το GPFS μπορεί να εκχωρεί κλειδώματα για μέρη του αρχείου μόνο αντί για ολόκληρο το αρχείο όπως γίνεται στο GFS. Αυτό έχει ως αποτέλεσμα στην περίπτωσή μας όπου όλοι οι κόμβοι πρέπει να προσπελάσουν δεδομένα ενός μοναδικού αρχείου, όταν χρησιμοποιείται το GPFS να μπορούν πολλοί κόμβοι να προσπελαύνουν ταυτόχρονα διαφορετικά μέρη του αρχείου. Αντίθετα όταν χρησιμοποιείται το GFS κάθε κόμβος που απαιτεί προσπέλαση στο αρχείο, πρέπει να πάρει κλείδωμα για ολόκληρο το αρχείο αναγκάζοντας τους άλλους κόμβους να περιμένουν ακόμα και αν έχουν σκοπό να προσπελάσουν διαφορετικό μέρος του αρχείου.

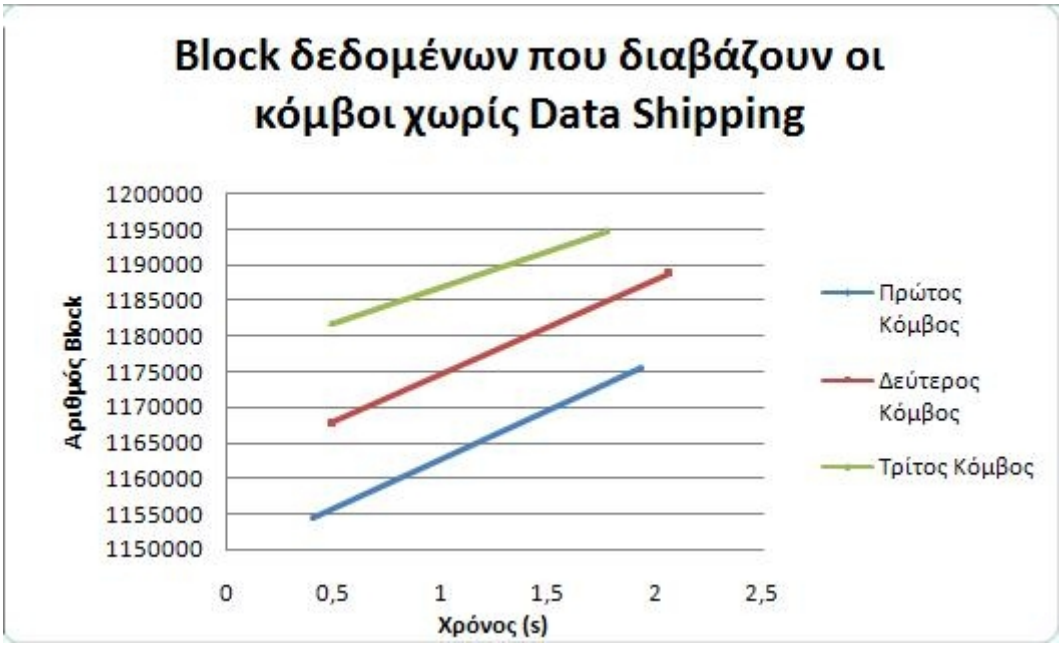
Μέρος 2^{ο}
Αρχικά γίνεται σύγκριση των καταγραφών που αντιστοιχούν στην παράλληλη ανάγνωση των τριών κόμβων με χρήση του gpfsperf χωρίς το Data Shipping Mode ενεργοποιημένο. Οι τρεις καταγραφές συγκρίνονται ανά δύο μεταξύ τους για να βρεθούν πόσα από τα δεδομένα είναι κοινά μεταξύ δύο κόμβων.
Στη συνέχεια κάνουμε την ίδια σύγκριση για τις καταγραφές που αντιστοιχούν στο ίδιο πείραμα αλλά με το Data Shipping Mode ενεργοποιημένο.
Ο παρακάτω πίνακας περιλαμβάνει τα αποτελέσματα της σύγκρισης. Η λογική του είναι η εξής: Τα δεδομένα των κελιών της διαγωνίου αντιστοιχούν στο πλήθος των blocks που περιλαμβάνει η καταγραφή του αντίστοιχου κόμβου. Τα δεδομένα των υπολοίπων κελιών δείχνουν τα blocks που εμφανίζονται στην καταγραφή του κόμβου Α και σε αυτή του κόμβου Β όπου ο κόμβος Α είναι αυτός που αναγράφεται στην τεταγμένη της καρτεσιανής συντεταγμένης του κελιού. Αντίστοιχα για τον κόμβο Β και την τετμημένη. Από την φύση των δεδομένων, αντιλαμβανόμαστε ο πίνακας είναι συμμετρικός.

	Κόμβος 1	Κόμβος 2	Κόμβος 3
Κόμβος 1	21539	7681	0
Κόμβος 2	7681	21010	7184
Κόμβος 3	0	7184	1330

Πίνακας 8.8: Πείραμα 5, Data Shipping

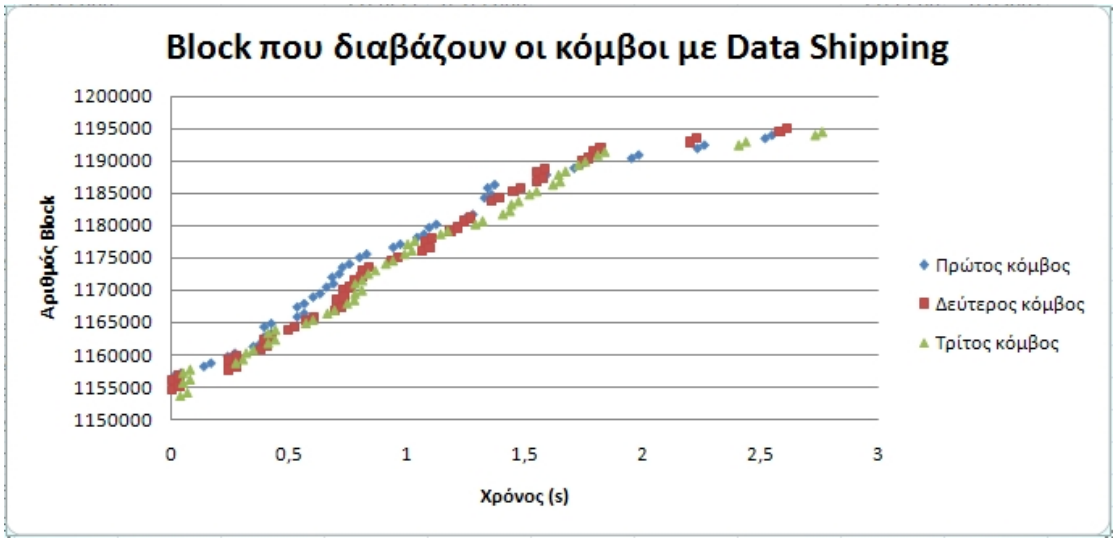
Τα αποτελέσματα που παίρνουμε για το Data Shipping Mode είναι τα απολύτως αναμενόμενα αφού βλέπουμε πως δεν υπάρχουν δύο καταγραφές που να περιέχουν κοινά blocks. Επιπλέον οι καταγραφές δείχνουν ότι όταν το Data Shipping είναι ενεργοποιημένο κάθε κόμβος δεν προσπελαύνει πολλά συνεχόμενα blocks όπως

γίνεται στις περιπτώσεις που το Data Shipping δεν είναι ενεργοποιημένο, αλλά κομμάτια 512 συνεχόμενων blocks.
Τα γεγονότα αυτά τα βλέπουμε και στις παρακάτω εικόνες. Σε αυτές έχουμε αναπαραστήσει γραφικά, τους αριθμούς block δεδομένων που προσπελαύνει κάθε κόμβος κατά τη διάρκεια της εκτέλεσης των πειραμάτων. Σε όλα τα διαγράμματα ο αριθμός block που προσπελάζονται εμφανίζονται στον Y άξονα, ενώ στον X άξονα υπάρχει ο χρόνος προσπέλασής τους. Ως χρόνος 0 θεωρείται ο χρόνος που έγινε η πρώτη προσπέλαση κάποιου block δεδομένων από οποιονδήποτε κόμβο.
Στην πρώτη εικόνα φαίνονται τα blocks δεδομένων που προσπελαύνουν οι κόμβοι όταν το Data Shipping(DS) Mode δεν ήταν ενεργοποιημένο. Στις επόμενες τρεις εικόνες αναπαριστούμε τα blocks δεδομένων που προσπέλασαν οι κόμβοι όταν το Data Shipping Mode ήταν ενεργοποιημένο. Ο λόγος που υπάρχουν τρία διαγράμματα για το ίδιο πείραμα είναι για να δειχθούν διαφορετικοί βαθμοί λεπτομέρειας.



Σχήμα 8.9: Blocks που διαβάζουν οι κόμβοι όταν δεν υπάρχει DS

Στην εικόνα αυτή βλέπουμε ξεκάθαρα ότι ο κόμβος 2 προσπελαύνει ορισμένα blocks που προσπελαύνουν ο κόμβος 1 και ορισμένα που προσπελαύνει ο κόμβος 3. Αντίθετα οι κόμβοι 1 και 3 δεν προσπελαύνουν κανένα κοινό block.
Δεν θα πρέπει να προξενεί εντύπωση ότι οι αριθμοί που προσδιορίζουν τα blocks, ξεκινάνε από 1150000 καθώς από αυτό το block του αποθηκευτικού χώρου και κάτω επέλεξε το GPFS να σώσει το αρχείο. Θα μπορούσε αυτός ο αριθμός να είναι οποιοσδήποτε άλλος, με τον περιορισμό να είναι μεγαλύτερος από το 0 και μικρότερος από το πλήθος των blocks του δίσκου πλην το μέγεθος του αρχείου σε blocks.
Επίσης πρέπει να σημειώσουμε ότι σε αυτή την περίπτωση έτυχε το αρχείο να αποθηκευτεί σε συνεχόμενα blocks. Θα μπορούσε το ίδιο εύκολα να είχαν επιλεχθεί από το GPFS πολλαπλά τμήματα του αποθηκευτικού χώρου για την αποθήκευση των δεδομένων του αρχείου.

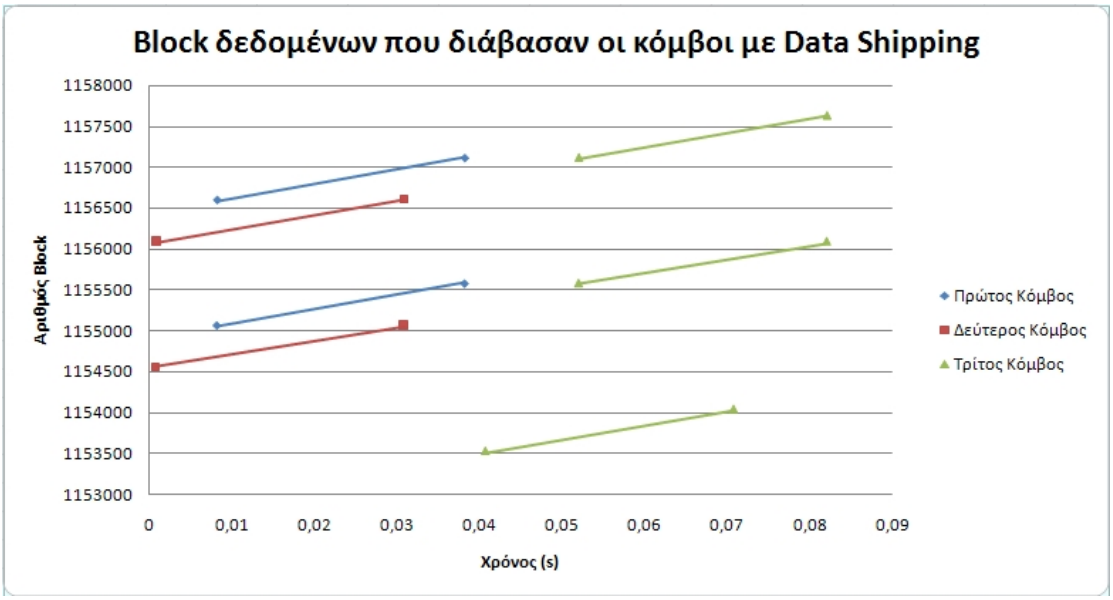


Σχήμα 8.10: Blocks που διαβάζουν οι κόμβοι όταν υπάρχει DS – Καθόλου λεπτομέρεια

Στην παραπάνω εικόνα παρατίθενται όλα τα blocks που προσπελαύνει κάθε κόμβος καθ' όλη τη διάρκεια του πειράματος. Από αυτή την εικόνα βλέπουμε γενικά ότι οι κόμβοι προσπελαύνουν διαφορετικά blocks αλλά ο μεγάλος αριθμός προσπελάσεων δεν μας επιτρέπει να δούμε λεπτομερώς το γεγονός αυτό.
Είναι προφανές από την εικόνα, ότι όσο περνάει ο χρόνος, οι κόμβοι διαβάζουν δεδομένα που αντιστοιχούν σε ολοένα και μεγαλύτερους αριθμούς blocks. Το γεγονός αυτό μας επιτρέπει να δούμε με μεγαλύτερη λεπτομέρεια ξεχωριστά ένα μέρος της καταγραφής χωρίς να υπάρχει κίνδυνος να αλλοιωθούν τα συμπεράσματα που προκύπτουν.
Ο συνολικός χρόνος ολοκλήρωσης των προσπελάσεων σε αυτή την εικόνα δεν θα πρέπει σε καμία περίπτωση να συγκριθεί με τον αντίστοιχο χρόνο στο προηγούμενο διάγραμμα με σκοπό την εξαγωγή συμπερασμάτων για την ταχύτητα των δύο ειδών προσπέλασης (με και χωρίς DS) . Όπως έχουμε πει , ο τρόπος που γίνεται η μελέτη μας δεν επιτρέπει την σύγκριση χρονικών αποτελεσμάτων των πειραμάτων.

Σχήμα 8.11: Blocks που διαβάζουν οι κόμβοι όταν υπάρχει DS – Μικρή λεπτομέρεια

Σε αυτή την εικόνα έχουμε κρατήσει μόνο τις προσβάσεις που έγιναν πριν από το χρόνο 0.8s. Όπως μπορεί να δει κανείς είναι αρκετά ξεκάθαρο ότι οι κόμβοι δεν προσπελαύνουν κοινά blocks.



Σχήμα 8.12: Blocks που διαβάζουν οι κόμβοι όταν υπάρχει DS – Μεγάλη λεπτομέρεια

Η εικόνα αυτή περιλαμβάνει μόνο τις προσπελάσεις στα blocks δεδομένων του αρχείου που έκαναν οι κόμβοι κατά το πρώτο 0.1s. Είναι απόλυτα ξεκάθαρο ότι όλοι οι κόμβοι προσπελαύνουν διαφορετικά blocks.

Μελετώντας και τις καταγραφές που παίρνουμε από το GPFS Trace, βλέπουμε ότι καταγράφονται πολλά DS γεγονότα, πράγμα που επιβεβαιώνει και τυπικά την ενεργοποίηση του Data Shipping Mode.

Κεφάλαιο 9.
Επίλογος

9.1 Σύνοψη

Στην παρούσα διπλωματική μελετήθηκαν, όσο της επέτρεπε ο σκοπός της, ορισμένα στοιχεία που αφορούν δύο συστήματα αρχείων, το GPFS και GFS με χρήση τεχνικών ιχνηλασίας (tracing). Το συστήματα αυτά αρχείων ανήκουν στην γενική κατηγορία των δικτυακών συστημάτων αρχείων και προορίζονται για περιβάλλοντα cluster. Προκειμένου να είναι δυνατή η μελέτη χωρίς την χρήση του απαραίτητου hardware, χρησιμοποιείται η τεχνολογία του Virtualization για την προσομοίωσή του.

Στα πρώτα κεφάλαια αναλύθηκε η βασική θεωρία που αφορά τον χειρισμό των συσκευών αποθήκευσης, των αρχείων και των συστημάτων αρχείων από τα σύγχρονα λειτουργικά συστήματα.

Στη συνέχεια δόθηκαν λεπτομέρειες για τον χειρισμό των συσκευών αποθήκευσης, των αρχείων και των συστημάτων αρχείων για το λειτουργικό σύστημα Linux, καθώς αυτό αποτέλεσε το λειτουργικό σύστημα στο οποίο πραγματοποιήθηκε η διπλωματική. Πρωτού φτάσουμε στο τελευταίο κεφάλαιο που αφορά την διεξαγωγή των πειραματικών μετρήσεων και την εξαγωγή συγκεκριμένων συμπερασμάτων, μεσολαβούν ορισμένα πολύ σημαντικά κεφάλαια. Σε αυτά αναλύονται διεξοδικά τα παράλληλα και τα συστήματα αρχείων κοινού μέσου, γίνεται μια σύγκριση των GFS και GPFS βασιζόμενη σε βιβλιογραφικά δεδομένα, περιγράφεται η δομή του συστήματος πάνω στο οποίο έγινε η μελέτη με αναφορά στα κύρια στοιχεία που χαρακτηρίζουν την τεχνολογία του Virtualization και τέλος αναλύονται οι τεχνικές tracing που εφαρμόζονται σήμερα στην ιεραρχία του λογισμικού που παρεμβάλλεται στην διεκπαιρέωση του παράλληλου I/O. Στο τελευταίο κεφάλαιο χρησιμοποιείται η το σύστημα που δημιουργήθηκε για την διενέργεια των πειραμάτων, έτσι ώστε κάνοντας χρήση ορισμένων από τις τεχνικές tracing, να προκύψουν χρήσιμα συμπεράσματα που αφορούν την λειτουργία των δύο υπό μελέτη συστημάτων αρχείων. Έμφαση δίνεται στην εξακρίβωση στοιχείων που αφορούν τον χειρισμό του caching των δύο συστημάτων αρχείων.

9.2 Επεκτάσεις

Η υποδομή που παρουσιάζεται και αναπτύσσεται σε αυτή την διπλωματική εργασία, δηλαδή η υποδομή που προκύπτει με την χρήση εικονικών μηχανών για την προσομοίωση του hardware που απαιτείται για την δημιουργία ενός περιβάλλοντος cluster, μπορεί να αποτελέσει την υποδομή πάνω στην οποία μπορούν να γίνουν μελλοντικές έρευνες προς τις παρακάτω κατευθύνσεις:

- Εκτενέστερη μελέτη των συστημάτων αρχείων GFS και GPFS.
- Μελέτη άλλων διαδεδομένων παράλληλων συστημάτων αρχείων όπως το Lustre και το PVFS2 και συστημάτων αρχείων κοινού μέσου όπως το OCFS2
- Χρήση των εργαλείων που αναλύθηκαν στο κεφάλαιο 7 που αφορούν το tracing στο επίπεδο της βιβλιοθήκης παράλληλου προγραμματισμού, ώστε να μελετηθεί εκτενώς η συμπεριφορά ορισμένων παράλληλων εφαρμογών. Από

την μελέτη της συμπεριφοράς των παράλληλων εφαρμογών μπορούν να προκύψουν στοιχεία που υποδεικνύουν την ύπαρξη bottlenecks και να χρησιμοποιηθούν ώστε να επανασχεδιαστεί η εφαρμογή με γνώμονα την επίτευξη υψηλότερων επιδόσεων.

Βιβλιογραφία

[Aag] Advanced Administration Guide, http://publib.boulder.ibm.com/epubs/pdf/bl1adv00.pdf

[Apr] Administration and Programming Reference, http://publib.boulder.ibm.com/epubs/pdf/bl1adm11.pdf

[ASP] ASC Purple, https://asc.llnl.gov/computing_resources/purple/

[Bar] James W. Barker, "A Survey of Clustered Parallel File Systems for High Performance Computing Clusters", http://rc.rit.edu/docs/Survey_of_Clustered_Parallel_File_Systems_004_LANL.ppt

[BC05] Daniel P. Bovet, Marco Cesati, “Understanding the Linux Kernel”, Third Edition, O'Reilly Media, 2005

[BGL] Blue Gene/L, https://asc.llnl.gov/computing_resources/bluegenel/

[Cfq] Cluster Project FAQ, http://sources.redhat.com/cluster/wiki/FAQ

[Cmc] Configuring and Managing a Red Hat Cluster for RHEL5.2, http://www.redhat.com/docs/enUS/Red_Hat_Enterprise_Linux/5.2/html/Cluster_Administration/index.html

[COH+05] Jason Cope, Michael Oberg, Henry M. Tufo and Matthew Woitaszek, "Shared Parallel Filesystems in Heterogeneous Linux Multi-Cluster Enviroments", Parallel Processing White Paper, 2005

[Cpi] Concepts, Planning, and Installation, http://publib.boulder.ibm.com/epubs/pdf/bl1ins11.pdf

[DWA+94] Michael D. Dahlin, Randolph Y. Wang, Thomas E. Anderson, David A. Patterson, ”Cooperative Caching: Using Remote Client Memory to Improve File System Performance”, Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation, 1994

[Exp] Exploring Clustered Parallel File Systems and Object Storage, http://softwarecommunity.intel.com/articles/eng/2640.htm

[Gfs] GFS 6.1 Administrator's Guide, http://www.redhat.com/docs/manuals/csgfs/pdf/rh-gfs-en-6_1.pdf

[Gnd] Using GNBD with GFS for RHEL5.2, http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.2/pdf/Global_Network_Block_Device/Global_Network_Block_Device.pdf

[Ifc] Introduction to Fibre Channel, http://www.cs.uml.edu/~bill/cs520/slides_11_fibre_channel.pdf

[Int] An introduction to GPFS, http://www03.ibm.com/systems/clusters/software/whitepapers/gpfs_intro.html

[Jat] Journal of Theoretical and Applied Information Technology, Comparison of Grid Computing vs. Cluster Computing, http://www.jatit.org/research/introduction_grid_computing.htm

[Kou] Kornilios-Antonios Kourtis, “Σχεδίαση και υλοποίηση μιας δικτυακής συσκευής block στο λειτουργικό σύστημα Linux” , Διπλωματική εργασία της σχολής ΗΜ-ΜΥ, ΕΜΠ

[LCC+05] Wei-keng Liao, Kenin Coloma, Alok Choudhary, Lee Ward, Eric Russell, and Sonja Tideman, "Collective Caching: Application-aware Client-side File Caching", Symposium on High Performance Distributed Computing, 2005

[Lcg] Linux Clusters vs. Grids, http://searchenterpriselinux.techtarget.com/tip/0,289483,sid39_gci1250726,00.html

[MCL+01] Shirley Moore, David Cronk, Kevin London, and Jack Dongarra "Review of Performance Analysis Tools for MPI Parallel Programs" , Proceedings of the 8th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, 2001

[Nan] Anastasios A. Nanos, "Σχεδίαση και Υλοποίηση Μηχανισμού Μεταφοράς Δεδομένων από Συσκευές Αποθήκευσης Myrinet, Χωρίς την Μεσολάβηση της Ιεραρχίας Μνήμης", Διπλωματική εργασία της σχολής ΗΜ-ΜΥ, ΕΜΠ

[Ocq] OCFS2 FAQ, http://oss.oracle.com/projects/ocfs2/dist/documentation/v1.2/ocfs2_faq.html

[Ocs] OCFS2 User Guide, http://oss.oracle.com/projects/ocfs2/dist/documentation/v1.2/ocfs2_users_guide.pdf

[Ofc] Fibre Channel · Overview of the Technology, http://www.fibrechannel.org/technology/overview.html

[P00] Kenneth W. Preslan, "Implementing Journaling in a Linux Shared Disk File System", Proceedings of the Eighth NASA Goddard Conference on Mass Storage Systems and Technologies in cooperation with the 17th IEEE Symposium on Mass Storage Systems, 2000

[PBE+99] Kenneth W. Preslan, Andrew P. Barry, Jonathan E. Brassow, Grant M. Erickson, Erling Nygaard, Christopher J. Sabol, Steven R. Soltis, David C. Teigland, and Matthew T. O'Keefe, "A 64-bit, Shared Disk File System for Linux", 16th IEEE Symposium on Mass Storage Systems, 1999

[Pdg] Problem Determination Guide, http://publib.boulder.ibm.com/epubs/pdf/bl1pdg11.pdf

[PH05] David A. Patterson, John L. Hennessy, "Computer Organization and Design, The Hardware/Software Interface", Third Edition, Morgan Kauffman Publishers, 2005

[Pvs] PVFS Home Page, http://www.pvfs.org/

[RCH05] Alessandro Rubini, Jonathan Corbet, Greg Kroah-Hartman, "Linux Device Drivers", Third Edition, O'Reilly Media, 2005

[Rcs] Red Hat Cluster Suite Overview for RHEL5.2, http://www.redhat.com/docs/enUS/Red_Hat_Enterprise_Linux/5.2/pdf/Cluster_Suite_Overview/Cluster_Suite_Overview.pdf

[Roc] Benjamic Roch, "Monolithic kernel vs. Microkernel"

[Scf] Supercomputing Facility for Bioinformatics & Computational Biology, IIT Delhi, Tutorial on Clustering, http://www.scfbio-iitd.res.in/doc/clustering.pdf

[Sfs] Cluster Technology and File Systems, http://www.communigate.com/notes/SFS.html

[SGG04] Abraham Silberschatz, Greg Gagne, Peter B. Galvin, "Operating System Concepts", Seventh Edition, John Wiley & Sons, 2004

[SH02] Frank Schmuck and Roger Hashkin, "GPFS : A Shared Disk File System for Large Computing Clusters", Proceedings of the FAST 2002

Conference on File and Storage Technologies
[SKS02] Abraham Silberschatz, Henry Korth, S. Sudarhan, “Database System Concepts”, Fourth Edition, McGraw-Hill, 2002

[Top] Top500, http://www.top500.org
[W07] Steven Whitehouse,"The GFS2 File System", Linux Symposium, 2007
[Xdc] Xen 3.0 Documentation, http://bits.xensource.com/Xen/docs/user.pdf
[Xen] An Overview of Xen Virtualization, http://www.dell.com/downloads/global/power/ps3q05-20050191-Abels.pdf